UNISYS

REPORTER III Vocabulary Language Operations Guide

Relative to Release
Level 2.1

Priced Item

January 1986

Printed in U S America
1177177

UNISYS

Product Information Announcement

○ New Release ○ Revision [✓ ● Update] ○ New Mail Code

Title

REPORTER III Vocabulary Language Operations Guide

This Product Information Announcement announces the release of Update 3 to the January 1985 publication of the *REPORTER III Vocabulary Language Operations Guide*. This update is relative to the A Series Mark 4.0.2 System Software Release, dated October 1993.

This update documents enhancements to REPORTER III to process and generate COBOL85 programs. These enhancements are being made to support users migrating from V Series systems to A Series systems.

Remove	**Insert**
	iiA through iiB
vii through viii	vii through viii
2–1 through 2–2	2–1 through 2–2
2–13 through 2–14	2–13 through 2–14
3–19 through 3–22	3–19 through 3–22
5–25 through 5–30	5–25 through 5–30
6–1 through 6–2	6–1 through 6–2
6–5 through 6–8	6–5 through 6–8
6–15 through 6–16	6–15 through 6–16
7–1 through 7–2	7–1 through 7–2
7–5 through 7–10	7–5 through 7–10
B–1 through B–2	B–1 through B–2
F–1 through F–2	F–1 through F–2
H–3 through H–8	H–3 through H–8
H–13 through H–16	H–13 through H–16
I–1 through I–2	I–1 through I–2
	I–5 through I–6
	J–1 through J–4

Changes are indicated by vertical bars in the margins of the replacement pages.

Retain this Product Information Announcement as a record of changes made to the base publication.

To order additional copies of this document

- United States customers, call Unisys Direct at 1-800-448-1424.
- All other customers, contact your Unisys Sales Office.
- Unisys personnel, use the Electronic Literature Ordering (ELO) system.

Announcement only:

Announcement and attachments: AS243

System: REPORTER III
Release: Mark 4.0.2 October 1993

Part number: 1177177–003

UNISYS

Publication Change Notice (PCN)

Date	Form—PCN number
10/25/88	1177177-002

Title REPORTER III Vocabulary Language Operations Guide (Relative to the Mark 2.4 System Software Release)

Description

This PCN provides revisions to the REPORTER III Vocabulary Language Operations Guide, relative to the 2.4 Software Release.

Revisions to the text are indicated by black vertical bars on the affected pages.

Replace These Pages	Add These Pages
iii thru vii	7-2A
7-1	
H-7	
H-17	

Retain the PCN cover sheet as a record of changes made to the basic publication.

Distribution Codes SC, SD, SE

UNISYS Publication Change Notice (PCN)

Date: January 20, 1986	Form—PCN number: 1177177-001
Title: REPORTER III Vocabulary Language User's Guide (January 1985)	

Description

This PCN provides revisions to the REPORTER III Vocabulary Language Users Guide, relative to the 2.1 Software Release. Revisions to the text are indicated by black vertical bars on the affected pages.

Replace These Pages	Add These Pages
4-33	4-62A
4-61	H-8A
4-63	
5-23	
5-27	
7-1	
7-5	
7-9	
7-13 thru 7-15	
7-19	
E-1	
H-5 thru H-7	
H-13 thru H-15	
1	
9 thru 11	

Retain this PCN cover sheet as a record of changes made to the basic publication.

UNISYS

PUBLICATIONS CENTER
BUILDING FOUR
41100 PLYMOUTH ROAD
PLYMOUTH, MICHIGAN
48170

January 17, 1985

REPORTER III
VOCABULARY LANGUAGE
USER'S MANUAL

Through our literature subscription services you have requested updates to the above manual. This manual has been revised to reflect level 2.0 software and is now available as form 1177177.

Your subscription has been changed to reflect the new number. If you want your subscription to revert to the old number (1149861), please notify the Literature Publication Center within 30 days.

We are sending you the number of copies of this new manual as specified in your subscription. If you do not want these manuals, please return them to the Literature Publication Center within 30 days.

Very truly yours,

Corporate Documentation

UNISYS

REPORTER III Vocabulary Language Operations Guide

Relative to Release
Level 2.1

Priced Item

January 1986

Printed in U S America
1177177

Correspondence regarding this publication should be forwarded, using the Product Improvement card at the back of this manual, or remarks may be addressed directly to Unisys Corporation, Production and Printing Services East, P.O. Box 500, Blue Bell, PA 19424 U.S.A.

Page Status

Page	Issue
iiA through iiB	–003
iii through vi	–002
vii through viii	–003
ix	–000
x	Blank
1–1 through 1–4	–000
2–1	–003
2–2 through 2–12	–000
2–13	–003
2–14 through 2–20	–000
3–1 through 3–19	–000
3–20 through 3–22	–003
3–23 through 3–25	–000
3–26	Blank
4–1 through 4–33	–000
4–34	–001
4–35 through 4–60	–000
4–61 through 4–62A	–001
4–62B	Blank
4–63	–001
4–64 through 4–69	–000
4–70	Blank
5–1 through 5–22	–000
5–23	–001
5–24	–000
5–25 through 5–26	–003
5–27	–001
5–28 through 5–30	–003
6–1	–003
6–2 through 6–5	–000
6–6	–003
6–7	–000
6–8	–003
6–9 through 6–15	–000
6–16	–003
6–17 through 6–18	–000

continued

continued

Page	Issue
7–1	–002
7–2	–003
7–2A	–002
7–2B	Blank
7–3 through 7–4	–000
7–5 through 7–9	–003
7–10 through 7–13	–000
7–14 through 7–15	–001
7–16 through 7–19	–000
7–20	–001
7–21 through 7–25	–000
7–26	Blank
A–1 through A–2	–000
B–1	–003
B–2 through B–4	–000
C–1 through C–5	–000
C–6	Blank
D–1 through D–4	–000
E–1	–000
E–2	–001
E–3	–000
E–4	Blank
F–1	–003
F–2	–000
G–1 through G–3	–000
G–4	Blank
H–1 through H–2	–000
H–3 through H–7	–003
H–8	–002
H–8A	–001
H–8B	Blank
H–9 through H–12	–000
H–13 through H–15	–003
H–16	–000
H–17 through H–18	–002
I–1 through I–2	–003
I–3 through I–4	–000
I–5	–003
I–6	Blank
J–1 through J–4	–003
1	–001
2 through 9	–000
10 through 12	–001

Unisys uses an 11-digit document numbering system. The suffix of the document number (1234 5678–*xyz*) indicates the document level. The first digit of the suffix (*x*) designates a revision level; the second digit (*y*) designates an update level. For example, the first release of a document has a suffix of –000. A suffix of –130 designates the third update to revision 1. The third digit (*z*) is used to indicate an errata for a particular level and is not reflected in the page status summary.

TABLE OF CONTENTS

INTRODUCTION

This guide defines the REPORTER III vocabulary language and describes its use with system files and Data Management Systems II (DMS II) data bases. The RP3VOC program creates a vocabulary of terms that describes data in the user data base. The report portion of the REPORTER III System uses the vocabulary to create printed reports and/or extract files.

Although this manual is directed primarily to the first-time or occasional vocabulary language user, more experienced users will find it a useful reference. To use it effectively, you should have a basic understanding of programming techniques and of the data base and its description in COBOL or DMS II. Some knowledge of COBOL data divisions is also desirable.

The contents of this manual are organized as follows:

Section 1	Section 1 is an overview of the vocabulary language, its capabilities, and its interface with the report language.
Section 2	Section 2 explains the basic language elements and constructs of the vocabulary language.
Section 3	Section 3 introduces vocabulary language statements. The section begins with a complete syntax diagram displaying all the language statements, and is accompanied by a table containing brief descriptions of each statement. The syntax diagram and table provide good references for the first-time user. The remainder of the section discusses general language statements.
Section 4	Section 4 discusses language statements for vocabularies accessing DMS II data bases. The section is organized in three subsections for B 1000, B 2000/B 3000/B 4000, and A Series of Systems.
Section 5	Section 5 discusses language statements for vocabularies accessing system files.

Section 6	Section 6 describes user-created input procedures.
Section 7	Section 7 is an operations guide for executing the RP3VOC program on A Series, B 1000, and B 2000/B 3000/B 4000 Series of Systems.
Appendices	Appendices A through I are references for system flow, defaults and limits, reserved words, error messages, and other related information. Appendix J provides information on converting A Series COBOL74 programs to COBOL85 programs.

The following manuals explain the use of the REPORTER III system and the optional On-Line REPORTER III module.

Operations Guide, REPORTER III Report Language (Relative to 2.4 Software Release), form 1177185.

Reference Card, REPORTER III (Relative to 2.0 Software Release), form 1177318.

Capabilities Manual, REPORTER III (Relative to 2.0 Software Release), form 1177300.

Operations Guide, On-Line REPORTER III (Relative to 2.0 Software Release), form 1177151.

Note that the "REPORTER III Report Language Operations Guide" was formerly titled "REPORTER III Report Language User's Guide" and that the "On-Line REPORTER III Operations Guide" was formerly titled "On-Line REPORTER III Report User's Guide."

The REPORTER III System is designed for use with the following:

1. A Series of Systems.
2. B 1000 Series of Systems.
3. B 2000/B 3000/B 4000 Series of Systems.
4. B 5000/B 6000/B 7000 Series of Systems.

Note that throughout this guide, instructions for A Series also pertain to B 5000/B 6000/B 7000 Series of Systems. Also note that when "COBOL" is used, it refers to the appropriate ANSI-74 COBOL for each system. If you are using an A Series System, the term "COBOL" also refers to ANSI-85 COBOL.

The style identification numbers for this product are A3 RP3 A9 RP3, B1000 RP3, B2000 RP3, B3000 RP3, B4000 RP3, B5000 RP3, B6000 RP3, and B7000 RP3.

SECTION 1

OVERVIEW OF THE VOCABULARY LANGUAGE

There are two processes of report production with the REPORTER III System. In the first process, you use the vocabulary language to describe the data files and/or data bases to be accessed during report production. The RP3VOC program processes the vocabulary language to produce two vocabulary files. These files are used in the second process to generate the report program.

VOCABULARY LANGUAGE CAPABILITIES

Using the vocabulary language, you can build specifications that create well-defined data items in reports. Using different vocabulary statements, you can rename, include, or exclude existing data items. These functions enable you to adapt vocabularies for unique report applications. You can name unidentified data items, or protect sensitive data from unauthorized users. You can perform these two functions with several different statements. You can also define shorthand terms for standard computations, keywords, or frequently used data names. These examples illustrate a few of the capabilities of the vocabulary language.

The vocabulary language is concise and English-like, and it creates vocabularies with minimal actual writing. One vocabulary can be used repeatedly to produce a variety of reports.

THE RP3VOC PROGRAM

The RP3VOC program processes the descriptions of system files and/or DMS II data bases and builds two vocabulary files containing all file names and data names, with descriptions such as size and relationship. The program allows you to specify the file descriptions to be included in the vocabulary, and to modify data names within the file descriptions.

The RP3VOC program can access data bases from combinations of the following data structures:

1. System files with indexed, relative, or sequential organization.

2. A Series of Systems DMS II data sets.

3. B 2000/B 3000/B 4000 Series of Systems DMS II data sets.

4. B 1000 Series of Systems DMS II data sets.

You can create your own COBOL data descriptions, or reference existing COBOL or DMS II record descriptions. RP3VOC accepts data stored on magnetic tape, disks, punched cards, or COBOL library files. Data descriptions vary for each data structure, and consist of:

1. System files - COBOL data names and file layouts in existing COBOL source programs. Data may also be described in COBOL notation and used as input directly to RP3VOC.

2. A Series DMS II data sets - descriptions of logical data sets, sets, and items specified in the Data and Structure Definition Language (DASDL) of A Series DMS II and stored in an A Series DMS II Data Base Description file.

3. B 2000/B 3000/B 4000 DMS II data sets- descriptions of logical data sets, sets, and items specified in the DASDL of B 2000/B 3000/B 4000 DMS II and stored in a B 2000/ B 3000/B 4000 DMS II Data Base Description file.

4. B 1000 DMS II data sets - descriptions of logical data sets, sets, and items specified in the DASDL of B 1000 DMS II and stored in the COBOL library files for the B 1000 DMS II disjoint data sets.

Any portion or all of an A Series or B 1000 - B 4000 Series DMS II data base can be included in a vocabulary. All related sets, access keys, and logical associations are automatically included along with A Series and B 1000 - B 4000 Series DMS II data sets.

RP3VOC analyzes vocabulary specifications, accesses the data descriptions, and prints a listing of the input specifications. If the specifications are invalid, error messages are displayed. Any errors are flagged with an appropriate error message, and printed out in the listing. The vocabulary files are built if the specifications are error-free.

The RP3VOC program generates two disk files, which are used as input for report production. A vocabulary file is generated that contains a directory of names, and corresponding characteristics. The other file, a COBOL library file, consists of COBOL source code describing files, records, and input procedures. The REPORTER III programs access the files to obtain the proper data file specifications. Output listings of the vocabulary are available on request. A more detailed explanation of

RP3VOC operation along with a system flow diagram is contained in Appendix A.

The following is a simple example of a vocabulary specification accessing a COBOL program source file for input. The necessary job control language statements have been added to show the minimum input required to produce the vocabulary. The control statements begin with a question mark (?).

```
?EXECUTE RP3VOC
?DATA VOCCRD
        VOCABULARY FILES ARE "VACCT" AND "VACCT2".
        LIST VOCABULARY.
        SOURCE FILE "ACCUPD" IS ON DISK.
        FILE ACCOUNT-MASTER.
        END FILE.
?END
```

The following are explanations of the individual specification statements:

VOCABULARY FILES ARE "VACCT" AND "VACCT2".

> The VOCABULARY statement names the two files that compose a vocabulary. The first external file name given, "VACCT", becomes the name of the vocabulary.

LIST VOCABULARY.

> The LIST statement requests a listing of the vocabulary output.

SOURCE FILE "ACCUPD" IS ON DISK.

> The SOURCE statement points RP3VOC at a COBOL source program on disk. The program, ACCUPD, is the update program for the master account file. This program contains the COBOL file and record descriptions for the ACCOUNT-MASTER file.

FILE ACCOUNT-MASTER.

> The FILE statement identifies the master account file, ACCOUNT-MASTER, to RP3VOC. RP3VOC locates the file and record descriptions in the COBOL source program, ACCUPD.

END FILE.

The END FILE statement informs RP3VOC that vocabulary specifications for the file, ACCOUNT-MASTER, are complete. RP3VOC performs the following two functions by default:

1. All data elements defined and named in the record descriptions are included in the vocabulary.

2. Editing pictures are created for each data element based on the item attributes (storage PICTURE) specified in the description of the record.

The output listing of the vocabulary is shown in Figure 1-1. This listing serves as the basis for report specifications.

```
RP3 N.NN          V O C A B U L A R Y  D I C T I O N A R Y          DD MMM YY
                                "VACCT"                                PAGE 1

SYSTEM FILES.

FILE ACCOUNT-MASTER.
     TOTAL-POPULATION IS DEFAULTED TO 9999.

LEVEL  NAME WITH QUALIFIERS  SUBSCRIPTS  TYPE         LENGTH  EDITING   STORAGE
                                                              PICTURE    LENGTH
  1    ACCOUNT-RECORD                    GROUP
  2    ACCT-NO                           ITEM NUMERIC    6    Z(6)          6
  2    BRANCH                            ITEM NUMERIC    4    Z(4)          4
  2    CUSTOMER-NAME                     ITEM STRING    20    X(20)        20
  2    CUSTOMER-ADDR                     GROUP
  3    STREET                            ITEM STRING    30    X(30)        30
  3    CITY-STATE-ZIP                    ITEM STRING    30    X(30)        30
  2    BALANCE-DUE                       ITEM NUMERIC    9   Z(5)9.99       8
  2    CREDIT-LIMIT                      ITEM NUMERIC    5    Z(5)          5
```

Figure 1-1. Vocabulary Listing, "VACCT"

SECTION 2

LANGUAGE ELEMENTS AND BASIC CONSTRUCTS

The vocabulary language is a high-level language based on English, and composed of characters, words, and statements. This section documents basic elements of the language, methods used to define vocabulary statements, and basic terminology used throughout the manual. In addition, this section discusses the basic language constructs used within many vocabulary statements; basic constructs include literals, numbers, strings, and COBOL pictures.

CHARACTER SETS

The vocabulary language character set consists of the following characters:

0-9	digits
A-Z	uppercase letters
a-z	lowercase letters*
*	asterisk or multiplication sign
@	at sign
	blank or space
[	bracket, left
]	bracket, right
:	colon
,	comma
$	dollar sign
=	equal sign
>	greater than symbol
<	less than symbol
-	minus sign or hyphen
#	number sign
(	parenthesis, left
)	parenthesis, right
%	percent sign
.	period or decimal point
+	plus sign
"	quotation mark
;	semi-colon
/	slash or division sign

* Lowercase letters can be used only by ANSI-85 COBOL users on A Series Systems.

SPECIFICATION FORM

The vocabulary language can be written on any COBOL-compatible coding form. You write the free-form statements in columns 8 through 72, using one or more spaces or appropriate punctuation to delimit elements of the language. Columns 1 through 6 are used for sequence numbers, and columns 73 through 80 are available for optional use, such as identification or remarks. The sequence number fields may be used for later updating of source language statements. Column 7 is not used and should be blank.

COMMENT INDICATOR

A percent sign (%) placed in columns 8 to 72 indicates that the characters that follow are part of a user comment. A percent sign placed between quotation marks ("%") is not interpreted as a comment. Comments are useful for documentation of the language specifications. They are not part of the vocabulary language and need not follow the rules of the language.

Examples:

```
% AN EXAMPLE OF THE USE OF COMMENTS.

VOCABULARY        % COMMENTS MAY EVEN
FILES ARE "V1"    % BE USED IN
AND "V2".         % A CONSTRUCT.
```

DEFINITION OF WORDS

A word is a combination of not more than 30 characters which may consist of the alphabetic characters A through Z, the numeric characters 0 through 9, and the hyphen (-) . A word must contain at least one alphabetic character. It cannot begin or end with a hyphen.

Vocabulary specifications are constructed with statements composed of vocabulary language words, symbols, and punctuation.

RESERVED WORDS

Reserved words are those words set aside in the vocabulary language which cannot be used as data names, file names, or procedure names. They include key words, optional words, and abbreviations. Refer to Appendix F for a list of vocabulary language reserved words, and to Appendix G for a list of report language reserved words.

Keywords

Another type of reserved word is a keyword. The category of keywords includes the words required to complete the meaning of statements and entries. The category also includes words that have a specific functional meaning. In the statement VOCABULARY FILES ARE "V1" AND "V2", the keywords are VOCABULARY and AND.

Optional Words

Optional words are reserved words included in the language to improve the readability of the statement formats. The optional words may be included or omitted by the user. For example, VOCABULARY FILES ARE "V1" AND "V2" is equivalent to VOCABULARY "V1" AND "V2"; thus, the inclusion or omission of the words FILES and ARE does not affect the logic of the statement.

NAMES

A name is a user-defined word. The name is defined as part of vocabulary by its existence in a referenced file or data-base description or its definition in the vocabulary specifications. In general, names should not be chosen which are identical to vocabulary or report language reserved words. In certain contexts within both of these languages, such names might be mistaken for reserved words. (Report language reserved words are given in Appendix G.)

METHOD OF LANGUAGE DEFINITION

The vocabulary language is defined by rules of syntax and semantics. The syntactic rules determine the structure of valid vocabulary-language statements. The semantics are rules that describe which vocabulary-language statements have valid meanings, and what those meanings are.

SYNTAX DIAGRAMS

The syntactical rules of the vocabulary language are described via a syntax diagram constructed of words and arrows. A syntactically correct statement is produced by tracing any path along the direction of the arrows. Words and symbols are written as they are encountered along the line paths. The syntax diagrams occasionally must be continued on a new line. In this case, the break in the arrows is shown by an on- or off-page connector.

Example:

```
  +<-- , <--+
  |         |
------>ROW------->THE------->BOAT----------------->DOWN-->(1)
          |       |       |       |
          +-->YOUR-->+       +-->GENTLY-->+

(1)------> - ------------------------------------->STREAM.
       |        |                     |
       |        |--------------->+    |
       |        |                |    |
       +-->THE--------->OLD-----> , ----->MILL---->|
                          |                        |
                          +----------------------->+
```

Valid productions from this syntax diagram include:

ROW THE BOAT DOWN - STREAM.
ROW, ROW, ROW YOUR BOAT GENTLY DOWN THE STREAM.
ROW, ROW, ROW, ROW THE BOAT DOWN THE OLD STREAM.
ROW YOUR BOAT DOWN THE MILL STREAM.
ROW THE BOAT DOWN THE OLD, MILL STREAM.

A bridge over a number indicates that the path may be traced up to the maximum number of times specified by the number under the bridge.

Example:

```
----->ACROSS--->THE---------------------->MISSOURI
                   |                  |
                   |<------  ,  <-----|
                   |                  |
                   |                  |
                   |--/ 1 /->BIG--->  |
                   |                  |
                   |                  |
                   |--/ 1 /->WIDE-->  |
                   |                  |
                   |                  |
                   +--/ 1 /->MUDDY->+
```

Valid productions include:

```
     ACROSS THE MISSOURI
     ACROSS THE BIG MISSOURI
     ACROSS THE MUDDY, WIDE MISSOURI
     ACROSS THE BIG, WIDE, MUDDY MISSOURI
```

but not:

```
     ACROSS THE BIG, BIG MISSOURI
     ACROSS THE WIDE, BIG, WIDE MISSOURI
```

A bridge over a number with an asterisk must be traced at least once, but not more than the maximum number of times specified by the number.

```
                     +<- , <--/ 1* /---+
                     |                 |
                     |                 |
------>ACROSS--->THE------/ 1 /->BIG---------------->MISSOURI
                     |                 |
                     |                 |
                     |--/ 1 /->WIDE--->|
                     |                 |
                     |                 |
                     +--/ 1 /->MUDDY-->+
```

Valid productions include:

ACROSS THE BIG, WIDE MISSOURI
ACROSS THE WIDE, MUDDY MISSOURI

but not:

ACROSS THE BIG MISSOURI
ACROSS THE BIG, WIDE, MUDDY MISSOURI
ACROSS THE BIG, BIG MISSOURI

Number bridges are usually used with loops, as in the preceding examples. Once a loop is exited, all number bridges within that loop are "reset" for the next entry into the loop.

Example:

```
        +<--------------------------------------- ,INTO<----------+
        |                                                          |
        |                                                          |
-->DOWN------>THE----------------------------/ 1 /->MISSOURI------>|
                      |                |   |                        |
                      |<---- , <-------|   |                        |
                      |                |   +---/ 1 /->SUWANEE------>|
                      |                |   |                        |
                      |--/ 1 /->BIG--->|   |                        |
                      |                |   |---/ 1 /->MISSISSIPPI-->|
                      |                |   |                        |
                      |--/ 1 /->WIDE-->|   |                        |
                      |                |   |---/ 1 /->OHIO--------->|
                      |                |   |                        |
                      +--/ 1 /->MUDDY->+   |                        |
                                           +---/ 1 /->GULF--------->+
```

Valid productions include:

DOWN THE MISSOURI
DOWN THE BIG OHIO
DOWN THE BIG OHIO, INTO THE BIG, WIDE MISSISSIPPI,
        INTO THE BIG, WIDE, MUDDY GULF

but not:

DOWN THE BIG OHIO, INTO THE MUDDY, MUDDY MISSISSIPPI
DOWN THE BIG OHIO, INTO THE MUDDY OHIO

RESERVED WORDS

Words in uppercase letters are the reserved words in the language and must be written exactly as shown in the syntax diagrams. Keywords are words required to complete the meaning of statements. Some key words may be abbreviated by omitting letters from the end of the word. The minimum required abbreviation is underlined in the syntax diagram. If a keyword must be used in its entirety, then the entire word is underlined. Optional words having no semantic meaning are included in the language to improve the readability of the statement formats. They are never underlined in the syntax diagrams and, if used, cannot be abbreviated.

Example:

```
------->VOCABULARY-------------------------------->(1)
                  |         | |           |
                  +->FILES->+ |->ARE---->|
                              |           |
                              +-> = ---->+

(1)--> <external file name> -->AND--> <external file name>-->.
```

The reserved words are VOCABULARY, FILES, ARE, and AND. VOCABULARY and AND are key words. VOCABULARY may be abbreviated as VOCAB, VOCABUL, VOCABULA, and so forth. FILES and ARE are optional words, and must be spelled out if they are used. The equal sign (=) is an optional symbol.

PUNCTUATION

Special characters such as the parenthesis, colon, period, and comma must be written as they appear in the syntax diagrams. Keywords, optional words, names, numerics, and character string constants must be separated from each other by a blank or a special character. Whenever a blank is used, several blanks may be given. A blank may also be placed around special characters for readability. A blank before a period is not required except when the period immediately follows a numeric literal.

SYNTACTIC VARIABLES

Phrases set off by angle brackets (<>) are syntactic variables which represent information to be supplied by you. A particular variable may represent a simple language element such as an integer, character, string, or name, or may represent a relatively complicated language construct such as a Boolean expression. These variables are defined either by verbal description or by syntax diagrams of their own. In either case, any valid language element or construct derived from the syntactic variable may be inserted into any diagram in place of the variable.

Example:

```
------>DOWN THE---> <river> --->INTO--->THE--> <ocean>
```

<river>:

```
----------->MISSISSIPPI--------------->
     |                  |
     |----->COLORADO------->|
     |                  |
     +----->HUDSON--------->+
```

<ocean>:

```
----------->ATLANTIC------------------>
     |                  |
     |----->PACIFIC-------->|
     |                  |
     +----->GULF----------->+
```

In the first diagram "river" and "ocean" are syntactic variables. Both are defined by the diagrams that follow. Syntactically valid productions include:

```
DOWN THE MISSISSIPPI INTO THE GULF
DOWN THE COLORADO INTO THE PACIFIC
DOWN THE COLORADO INTO THE GULF
DOWN THE MISSISSIPPI INTO THE PACIFIC
```

<u>SEMANTIC</u> <u>RULES</u>

Semantic rules are linked to the syntax diagrams by means of letters which label the critical paths. The letters reference paragraphs which explain the meaning associated with the corresponding syntax path, and explain additional rules associated with a choice in paths. These rules may make certain syntactically correct statements invalid.

Example:

```
--->DOWN-->THE---> <river> --->INTO-->THE---> <ocean>
```

<river>:

```
               A
-------------->MISSISSIPPI----------->
          |                 |
          | B               |
          |--->COLORADO---->|
          |                 |
          | C               |
          +--->HUDSON------>+
```

<ocean>:

```
               A
-------------->ATLANTIC-------------->
          |                 |
          | B               |
          +--->PACIFIC----->+
```

They are written so that they prompt you to choose the correct paths and supply the required information. By tracing the syntax diagram paths and reading the "path prompts," you compose the vocabulary specifications which describe the application.

The semantic rules for "river" would be explained in accompanying paragraphs labeled A, B, C. The semantic rules for "ocean" would be given in accompanying paragraphs labeled A and B. These paragraphs would explain that the choice of river must correctly match the choice of ocean so that the river flows into the correct ocean. Semantically valid productions include:

```
DOWN THE MISSISSIPPI INTO THE ATLANTIC
DOWN THE HUDSON INTO THE ATLANTIC
DOWN THE COLORADO INTO THE PACIFIC
```

but not:

```
DOWN THE MISSISSIPPI INTO THE PACIFIC
DOWN THE COLORADO INTO THE ATLANTIC
```

BASIC LANGUAGE CONSTRUCTS

Vocabulary language basic constructs are used within many vocabulary statements. They are literals, COBOL pictures, name qualification, and external file names.

LITERALS

A literal represents an element which has a value identical to the value being described. There are two classes of literals: numeric literals such as integers or numbers, and non-numeric literals or strings.

Numbers

A number is defined as an element composed of the digits 0 through 9, the plus sign (+) or the minus sign (-), and the decimal point. The rules for the formation of a number are:

1. Only one sign character and one decimal point may be contained in a number.

2. A number must contain at least one digit.

3. The sign of a number must appear as the leftmost character. If a sign is not present, the number is defined as a positive value.

4. The decimal point may appear anywhere within the number except the rightmost character. The absence of a decimal point denotes an integer.

Examples:

The forms 25.0 or 25 are correct.
The form 25. is incorrect.

5. A number cannot exceed 18 characters for B 1000, B 2000/B 3000/B 4000 or 22 characters for A Series systems, including the optional sign (+ or -) character. A number cannot have more than six digits to the right of the decimal point. The following are examples of numbers:

   13247 .005 +1.808 -.0968 7894.64 10

Integers

An integer is a number which contains no decimal point.

Strings

A string can contain any allowable character with the exception of the quotation mark. The beginning and end of the string is denoted by a quotation mark and any character enclosed by the quotation marks is a part of the string. A maximum of 55 characters can be used in a string.

Examples:

```
"THIS IS A SAMPLE STRING  #/$."
"ACTUAL SALES FIGURE"
"1234.567"
```

COBOL PICTURE

A COBOL picture is optionally used to describe how an item is to be printed. A COBOL picture, when specified as a string in the vocabulary, must have no blanks after the first quote and no blanks before the ending quote. It must be a valid COBOL editing picture. It is taken as given and associated with a particular item in the vocabulary. That is, no syntax checking is performed by the RP3VOC program. If the option DECIMAL-POINT IS COMMA has been set, the correct transposition of commas and decimal points must be given.

The following characters are used in forming editing pictures:

A	alpha character position
X	alphanumeric position
.	actual decimal point (or comma (,) if DECIMAL-POINT IS COMMA set)
*	asterisk (check protect)
,	comma (or actual decimal position if DECIMAL-POINT IS COMMA set)
CR	credit
DB	debit
9	digit position
$	dollar sign
I	insertion indicator (B 2000-B 4000 Systems only)
/	insertion of "/" character
-	minus
+	plus
B	space
0	zero
Z	zero suppress

An ANSI-74 COBOL or an ANSI-85 COBOL reference manual should be consulted for the exact rules for forming COBOL editing pictures. Examples:

Without the option DECIMAL-POINT IS COMMA	With the option DECIMAL-POINT IS COMMA
"$(5)Z.99"	"$(5)Z,99"
"ZZZ,ZZZ.99"	"ZZZ.ZZZ,99"
"X(10)"	"X(10)"
"Z(5)"	"Z(5)"
"99/99/99"	"99/99/99"

When reporting a vocabulary item, the REPORTER III System uses the editing picture available in the vocabulary. You can either specify the editing picture using WITH PICTURE clause, or for DMS II, the <ASSIGN statement>; or you can allow RP3VOC to build default editing pictures from defined storage pictures.

The conventions used by RP3VOC to generate default output editing pictures are:

1. Only numeric pictures are changed.

2. The assumed decimal point, if present, is changed to a period unless DECIMAL-POINT IS COMMA is set, in which case it is changed to a comma.

3. The picture symbol 9 is changed to a Z unless it is the least significant character before the decimal point or unless it follows the decimal point.

4. The picture symbols S, J, and K are changed to a minus sign (-) when found as the first character of the picture.

5. For A Series DMSII items, the COBOL picture representation is the basis for default editing pictures, with the exception of BOOLEAN and FIELD items. BOOLEAN items are assigned a default editing picture of X(5) for printing TRUE or FALSE. FIELD items are assigned a default editing picture appropriate to print the decimal equivalent of the binary field.

The following examples illustrate the default actions taken by RP3VOC when no output picture is specified. Implied pictures are depicted without the option DECIMAL POINT IS COMMA and with the option DECIMAL POINT IS COMMA.

Input Picture	Without Option	With Option
A(5)	No change	No change
XXX	No change	No change
99	Z9	Z9
9(5)V99	Z(5).99	Z(5),99
V99	.99	,99
9(3)	Z(3)	Z(3)
9(4)9V99	Z(4)9.99	Z(4)9,99
9V9(5)	9.9(5)	9,9(5)
S9(5)V99	-Z(5).99	-Z(5).99

NAME QUALIFICATION

Use name qualification to distinguish between data items or elements with the same name. RP3VOC cannot differentiate two data items with the same name. You need to qualify each identical data item name you want included in your vocabulary.

Qualifiers are any higher level items that uniquely identify the data item or element. For example, in a COBOL source file, you would use a group or record name as a qualifier for a data item. For a DMS II file, you would use a data set name to qualify a data element. You must use as many qualifiers as necessary to uniquely define data.

RP3VOC accesses data in COBOL source or DMS II data bases sequentially. In the following sample code, note that two data items in two different records have the same name.

```
01 INVENTORY
   02 PART-RECORD
     03 PART-NUMBER


01 PRODUCTION
   02 PART-RECORD
     03 PART-NUMBER
```

Note that the <qualifier> must precede the <name> item in the COBOL source code. For example, INVENTORY must precede PART-RECORD.

The following statements illustrate how you would qualify the data items:

```
PART-NUMBER OF PART-RECORD IN INVENTORY
PART-NUMBER OF PART-RECORD OF PRODUCTION
```

However, it is only necessary to give the next higher level which will uniquely qualify the name. The following statements show the minimum needed to qualify the data items:

```
PART-NUMBER OF INVENTORY
PART-NUMBER OF PRODUCTION
```

The syntax for name qualification is as follows:

```
                  +<--------------------------+
                  |                          E|
 A                | B        D                | F
--> <data name> ------>OF-----> <qualifier> --------->
                   |      |
                   | C    |
                   +-->IN->+
```

The paths of this syntax diagram follow.

Path	Explanation
A	The data item name being qualified is indicated here. Example: `NAME OF PERSONNEL-RECORD` In this example, NAME is the data item name that requires qualification.
B-C	OF and IN are synonymous.
D	Indicate the name <qualifier> here. Example: `ADDR1 OF ADDRESS OF PERSONNEL-RECORD` In this example, ADDRESS and PERSONNEL-RECORD are the <qualifier>s.
E	Take this path as often as required to specify additional <qualifier>s.
F	Take this path when you have finished specifying <qualifier>s.

EXTERNAL FILE NAME

An external file name identifies a file to the MCP. It consists of one or more strings separated by slashes with an optional disk pack (cartridge) clause.

A Series of Systems

An external file name consists of a series of <identifier>s separated by slashes and enclosed in quotation marks: file <identifier>, volume <identifier>, and file directory <identifier>s. An <identifier> may contain a maximum of 17 characters. Excluding the optional ON clause, a maximum of 30 characters is allowed for and external file name, including slashes. The syntax is as follows:

```
     A                          C
--- " ----------------------------> <identifier>" --> (1)
      |                          |
      |                          |
      |<-------------------------|
      |                          |
      | B                        |
      +---> <identifier>/ -->+

        D
(1) --------------------------------------------------->
      | E                                 |
      |          F                        |
      +------------> ON "<identifier>" --->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if you do not require a volume <identifier> or file directory <identifier>s.
B	Take this path to supply optional file directory <identifier>s and a volume <identifier>.
C	This <identifier> represents the file name.
D	Take this path if you do not specify a pack name.

Examples:

"VOCAB1"
"SAVINGS/NAME/AND/ADDRESS"
"A/B/C/D/E/F"

E   Take this path to specify a pack name.

F   The <identifier> represents a pack name or family name.

Examples:

"PERSONNEL/FILE" ON "EMPLOYEES"

B 1000 Series of Systems

An external file name contains a maximum of three <identifier>s. An <identifier> may have a maximum of 10 characters, and generally contains no special characters. The maximum size of an external file name, including all identifiers and slashes, is 28 characters. The syntax is as follows:

```
         A                         C
-- " -------------------------------> <identifier>" -> (1)
         |                         |
         | B                       |
         +--> <identifier>/   ->+

         D
(1)---------------------------------------------------------->
           |                                        |
           | E                                      |
           |-------------->ON  "<identifier>" -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if you do not supply a family name.
B	This <identifier> represents the family name.

C   This <identifier> represents the file name.

D   Take this path if you do not specify a pack ID.

Example:

```
"VOCAB1"
"PAYROLL10/TSTVCB"
```

E   This <identifier> represents a pack ID.

Example:

```
"VOCAB" ON "SYSTEMFILE"
"EMPLOYEES" ON "PAYROLL74"
```

<u>B 2000/B 3000/B 4000 Series of Systems</u>

An external file name contains one or two <identifier>s. An <identifier> may contain a maximum of six characters. You cannot use special characters (space, comma, period, slash, hyphen, and semicolon). The <identifier> specified here is the file name. The syntax is as follows:

```
 A                       B
---> "<identifier>" ------------------------------------------->
                       | C                                  |
                       |          D                         |
                       +---------->ON "<identifier>" ---->+
```

The paths of this syntax diagram are explained below:

<u>Path</u>	<u>Explanation</u>
A	Indicate the file name <identifier> here.
B	Take this path if you do not specify a pack ID.

Examples:

```
"VOCAB1"
"EV1"
"12JA74"
```

C Take this path to specify a pack ID.

D The <identifier> represents a pack ID.

Examples:

```
"STAFF" ON "SYSTEM"
```

SECTION 3

GENERAL LANGUAGE STATEMENTS

This section is the first to discuss REPORTER III Vocabulary Language statements. In this section, you will be introduced to the vocabulary language by a complete syntax diagram displaying all the language statements. Accompanying the diagram is a table containing brief descriptions of each statement.

Statements in the syntax diagram are organized into four groups:

1. General statements.
2. DMS II statements.
3. System file statements.
4. Input procedure.

General statements have the same syntax and function on all systems. Later in this section, each general statement has a detailed explanation along with a detailed syntax diagram.

DMS II statements let you modify and add data items from DMS II data bases to your vocabulary. The functions of these statements vary slightly from system to system. DMS II statements are described in three subsections for A Series, B 1000, and B 2000/B 3000/B 4000 Series of Systems in Section 4.

System file statements allow you to modify and add system files to your vocabulary. The function of system file statements are similiar on all systems. System files are discussed in Section 5.

When input routines generated by the report program cannot handle your needs, you can use input procedures to create unique input methods. The function of input procedures are similiar on all systems. Input procedures are discussed in Section 6.

The four groups listed above show the order that statements must appear if used in a specification. For example, general statements must always precede DMS II statements in a specification; DMS II statements must precede system file statements; system file statements must precede input procedures.

Statements are listed alphabetically in the syntax diagram, but you are not required to use the statements in that order in a specification. For example, when you use DMS II statements, the ASSIGN statement does not have to precede the DATASET statement in a specification.

The general syntax for the REPORTER III Vocabulary Language is as follows:

<u>General statements:</u>

```
---------> <VOCABULARY statement> --------------> (1)

(1)----------------------------------------------> (2)
        |                              |
        |<-----------------------------|
        |                              |
        |--> <LIST statement> -------->|
        |                              |
        |--> <PASSWORD statement> ---->|
        |                              |
        |--> <PURGE statement> ------->|
        |                              |
        |--> <SET option statement> -->|
        |                              |
        +--> <USER NAME statement>  -->+

(2) ---------------------------------------------------------------> (3)
        |                                                      |
        |<-----------------------------------------------------|
        |                                                      |
        +--> <REPLACE statement> ----------------------------->|
                                    |                          |
                                    +--> <COMMENT statement> -->+
```

DMS II statements:

```
(3) --------------------------------------------------------------------> (4)
       |                                                              |
       |<-------------------------------------------------------------|
       |                                                              |
       +------> <DB          -----------------------------> <END DB    ->+
                 statement> |                 |            statement>
                            |<----------------|
                            |                 |
                            |-> <ASSIGN    -->|
                            |    statement>   |
                            |                 |
                            |-> <DATASET   -->|
                            |    statement>   |
                            |                 |
                            |-> <EXCLUDE   -->|
                            |    statement>   |
                            |                 |
                            |-> <REDEFINE -->|
                            |    statement>   |
                            |                 |
                            +-> *<SET      -->+
                                  statement>
```

*NOTE: The <SET statement> is not available for B 1000 users.

System file statements:

```
(4) ---------------------------------------------------------------> (5)
       |
       |<--------------------------------------------------------- (6)
       |
       +------> <SOURCE statement> --------------------------------> (7)
                                       |                    |
                                       +-> <COBOL SOURCE> ->+

(5) ---------------------------------------------------------------> (8)
                                                             |
(6) <--------------------------------------------------------|
                                                             |
(7) --> <FILE      ------------------------------> <END FILE ->+
     statement> |                          |      statement>
                |<-------------------------|
                |                          |
                |-> <DATA-NAME-CHANGE ---->|
                |    statement>            |
                |                          |
                +-> <CONDITION-ADDITION ->+
                     statement>
```

Input Procedure:

```
(8) -----------------------------------------------------------> (9)
       |
       |<----------------------------------------------------- (10)
       |
       +--> <INPUT PROCEDURE --> <COBOL code for input ------> (11)
              statement>           procedure>

(9) ----------------------------------------------------------->
                                                           |
(10) <-----------------------------------------------------|
                                                           |
(11) ------------------------------> <END PROCEDURE -->+
        |                          |   statement>
        +--> <WORKING-STORAGE ---->+
               specification>
```

The following table provides brief descriptions of each statement shown in the preceding syntax diagram. The statements are listed in alphabetical order.

Language Statement	Explanation
ASSIGN Statement	Use the ASSIGN statement to specify an editing picture for an elementary data item. This picture is used when reporting the data item.
COBOL CODE FOR INPUT PROCEDURE	The COBOL CODE FOR INPUT PROCEDURE is the main element of an input procedure. It is the actual COBOL source code that accesses the data.
CONDITION-ADDITION Statement	Use the CONDITION-ADDITION statement to create a condition to describe an existing COBOL data item.
DATA-NAME-CHANGE Statement	Use the DATA-NAME-CHANGE statement to change COBOL data-names and/or assign new editing pictures. These pictures are used when reporting the data item.
DATASET Statement	Use this statement three ways: 1. To override the default total population for a data set. 2. To give an alternate name to a disjoint data set. 3. To redescribe a disjoint data set using the REDEFINE statement. B 1000 users must use the DATASET statement to add a disjoint data set to the vocabulary.
DB Statement	Use the DB statement to specify the physical or logical data base to be included in the vocabulary.

END DB Statement

Every DB statement must have a corresponding END DB statement. The END DB statement signals the end of specifications for the specified data base.

For A Series and B 2000 through B 4000 systems, this statement tells RP3VOC to automatically add any remaining elements of the DB INVOKE description to the vocabulary.

For the B 1000 Series of Systems, this statement tells RP3VOC to add the remaining elements of the current data set COBOL library file to the vocabulary.

END PROCEDURE Statement

The END PROCEDURE statement signals the end of the input procedure. Each INPUT PROCEDURE statement must have a corresponding END PROCEDURE statement.

EXCLUDE Statement

Use the EXCLUDE statement to exclude elements in disjoint data sets. For A Series and B 2000 through B 4000 systems, you can also exclude an entire data set.

FILE Statement

Use the FILE statement to identify the source file you want added to the vocabulary. You can also use this statement to change the internal and external file names of the file.

INPUT PROCEDURE Statement

Use the INPUT PROCEDURE statement to instruct RP3VOC that the subsequent COBOL code is part of the input procedure.

You can also use this statement to access system files and/or DMS II data bases previously defined in the vocabulary specification.

LIST Statement

Use the LIST statement to obtain various output listings.

PASSWORD Statement	Use the PASSWORD statement to establish security for your vocabulary. Only individuals using the specified password can access the vocabulary.
PURGE Statement	Use the PURGE statement to remove the RP3VOC specification after it has been processed.
REDEFINE Statement	Use the REDEFINE statement to reenter a disjoint data set into the vocabulary under a different name. This statement allows you to have the capability of multiple work areas for a data set in the generated report program.
REPLACE Statement	Use the REPLACE statement to define a name which is replaced by text during the report phase. This statement allows you to use a convenient shorthand technique for expanding the report language.
SET Statement	Use the SET statement to specify an alternate name for a disjoint data set. B 1000 users cannot use this statement.
SET OPTION Statement	The SET OPTION statement allows you to override default blocking values assigned to the two internal work files generated by the report program. This statement affects all reports using the specific vocabulary unless you override it in the report specifications.
SOURCE Statement	Use the SOURCE statement to identify the COBOL source containing the file and/or record descriptions you want to add to the vocabulary. This statement allows you to alternately add the COBOL source in-line with the vocabulary statements.
USER NAME Statement	Use the USER NAME statement for documentation purposes only. This statement allows you to identify the "owner" or creator of the vocabulary.

WORKING-STORAGE Specification	Use the WORKING-STORAGE specification to modify record descriptions in the WORKING-STORAGE SECTION of the COBOL CODE FOR INPUT PROCEDURE.
VOCABULARY Statement	Use the VOCABULARY statement to assign names to the two files (the vocabulary library and vocabulary names files) that are created by RP3VOC. You must use this statement as the first line of your vocabulary. There are no defaults for this statement.

Individual syntax diagrams and descriptions for general language statements are described below. The statements are presented in alphabetical order.

COMMENT STATEMENT

Use the COMMENT statement to add explanations of macro names to the vocabulary. These macro names must be previously defined with a REPLACE statement. You cannot use a COMMENT statement without a REPLACE statement. The comments you enter are printed in the vocabulary listing if you specify VOCABULARY in the LIST statement. The syntax for the COMMENT statement is as follows:

```
------>COMMENTS:  <comment text>  END-COMMENTS.
```

The <comment text> is any string of allowable characters starting in column 8 and ending in column 72. RP3VOC interprets as comment text any text immediately following the colon after "COMMENTS," and ending before "END-COMMENTS." The key word "END-COMMENTS" must be the first entity on a line.

You can enter an unlimited amount of text as comment text. This text appears in the vocabulary listing exactly as you specify it. The comments are printed immediately after the associated vocabulary element. Each line image is preceded by a percent sign (%) and a space on the listing.

Example:

```
     REPLACE ACCT-CUST-INFO BY ACCTS-RECV,
        CUST-INFO FROM ACCT-RECV VIA NUMBER-SET AT
        CUSTOMER-NUMBER = CUST-NO #.

     REPLACE ACCT-INVOICE-INFO BY ACCTS-RECV
        (INVOICE-INFO VIA INVOICES)#.


     COMMENTS:
          USAGE: ACCT-INVOICE-INFO SHOULD BE USED
          IN THE INPUT STATEMENT AS FOLLOWS:
            INPUT ACCT-INVOICE-INFO.

            THIS WILL ALLOW REFERENCE IN SUBSEQUENT REPORT STATEMENTS
            TO ALL INFORMATION ABOUT EACH INVOICE AND ALL INFORMATION
            ABOUT EACH RELATED ACCOUNTS RECEIVABLE RECORD.


     END-COM.
         .
         .
         .
```

The comments for the macro ACCT-INVOICE-INFO appear in the vocabulary listing as shown in Figure 3-1.

```
 RP3 N.NN              V O C A B U L A R Y             DD MMM YY
                            "CLIENT"                   PAGE 1

 PERMANENT MACROS (REPLACE STATEMENTS).

MACRO ACCT-CUST-INFO.
         TEXT:  ACCTS-RECV, CUST-INFO FROM ACCT-RECV VIA NUMBER-SET
                 AT CUSTOMR-NUMBER = CUST-NO #

MACRO ACCT-INVOICE-INFO.
         TEXT:  ACCTS-RECV(INVOICE-INFO VIA INVOICES) #
%       USAGE:  ACCTS-INVOICE-INFO SHOULD BE USED IN THE INPUT
%                STATEMENT AS FOLLOWS:
%
%                       INPUT ACCT-INVOICE-INFO.
%
%                THIS WILL ALLOW REFERENCE IN SUBSEQUENT
%                REPORTER STATEMENTS TO ALL INFORMATION ABOUT EACH
%                RELATED ACCOUNTS RECEIVABLE RECORD.
            .
            .
            .
```

Figure 3-1. Vocabulary Listing with Comment Text

LIST STATEMENT

Use the LIST statement to obtain output listings for a new or already existing vocabulary. RP3VOC automatically produces a listing of the vocabulary specifications, independent of a LIST statement. The syntax for the LIST statement is as follows:

```
      +<-----------------+
      | |      |        E|
      | +<-- , <--+      |
      |                  |
      |    A             |   F
-->LIST-------->LIBRARY--------------------------------> .
          |            |   |          |
          | B          |   | G        |
          |-->DICTIONARY--->|   +-->NOTES---->+
          |            |
          | C          |
          |-->SYNTAX------->|
          |            |
          | D          |
          +-->GO----------->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if you want a listing of the VOCABULARY LIBRARY file. This file contains the COBOL source code for file descriptions of system files if any were input to the vocabulary. This file also contains the COBOL source code for input procedures if any were input to the vocabulary.
B	Take this path to produce a listing of the VOCABULARY DICTIONARY file. This file contains information about data names that were input to the vocabulary. The listing appears in the following order: 1. Permanent macros (REPLACE Statement). 2. DMSII data bases. 3. System files. 4. Input procedures.

C Use this option to produce a COBOL syntax compile of the items placed into the vocabulary files by RP3VOC. Any errors in COBOL source code at this level result in syntax errors in the generated report programs. We strongly recommend that you use this option.

See Section 7 for important operations instructions regarding the syntax compile.

D Take this path if your vocabulary contains B 1000 Series DMSII data bases. With this option, RP3VOC may validate the physical- and/or logical-data-base names declared in the vocabulary specification.

For each data base, this option produces a COBOL skeleton program which is compiled and executed to invoke, open, and close each data base. For the program to execute successfully, the data base must be present.

The GO option overrides the action of the SYNTAX option on DMS II data bases.

E Take this path to specify additional options.

F Take this path after you have specified all list options.

G NOTES, in combination with the other options, uses an existing vocabulary to obtain the requested listings. After this option, no further vocabulary specifications are processed.

Example:

```
LIST LIBRARY DICTIONARY SYNTAX NOTES.
```

PASSWORD STATEMENT

The PASSWORD statement provides vocabulary security so that only individuals using the specified password can access the vocabulary. Only report specifications which contain this password may use the vocabulary. The syntax for the PASSWORD statement is as follows:

```
                                          A
--->PASSWORD--------------------> <literal> -------> .
                |          | |                    |
                |-->IS---->| | B                  |
                |          | +---> <word> ----->+
                +--> = --->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A-B	To specify a <literal> for your password, use path A. To specify a <word>, use path B.

The password must not exceed 10 characters. In report specifications, use this password exactly as described in the PASSWORD statement, including blanks and special characters.

Examples:

```
PASSWORD= "HELLO".
PASSWORD= "012/*".
PASSWORD PAYROLL.
PASSWORD IS EXTRA-LONG.
```

PURGE STATEMENT

Use the PURGE statement to automatically remove specifications after they are processed by RP3VOC. The PURGE option is generally used for specifications residing on disk.

To use vocabulary specifications residing on a disk file, you must file-equate. On B 2000/B 3000/B 4000 Series of Systems, you must also execute RP3VOC with a VALUE statement. Refer to the appropriate operations instructions in Section 7 of this manual for details.

The syntax for the PURGE statement is as follows:

```
------->PURGE---------------->SPECIFICATIONS.
```

REPLACE STATEMENT

Use the REPLACE statement to define a <name> that represents a portion of language <text> consisting of characters, words, or phrases. You can use this name to reference the <text> in report specifications which use the vocabulary. The REPLACE statement gives you a convenient shorthand technique for expanding the report language. The syntax for the REPLACE statement is as follows:

```
---->REPLACE-------------------------------------------------> (1)
          |         | |                 | |        |
          +-->ALL-->+ +-->OCCURRENCES-->+ +-->OF-->+


     +<----------------------------------- , <-------------- (2)
     |
   A |              B
(1) ----> <name> -------------------------------------------> (3)
                  |                                      |
                  |                                      |
                  |                                      |
                  |                                      |
                  |         +<--/ 9 /-- , <---+          |
                  |         |              E |           |
                  |         |                |           |
                  | C       | D              | F         |
                  +--> ( -----> <parameter> -----> ) ->+


(2) <---------------------------+
                              H |
           G                    | I
(3) --->BY---> <text> --> # --------> .
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Use this path to define a <name> that represents the <text> of the REPLACE statement. This <name> may then be used in the report language specifications to reference the text. The <name> is also called a macro name. Choose a unique word for the macro name. If you use a <name> that is the same as a report specification construct, you will lose the usage of that construct.

Example:

```
     REPLACE ALL OCCURRENCES OF TIMES BY *#.
```

This statement defines "TIMES" as the macro name that represents the text "*". The following statement is written in the report specifications:

```
     TOTAL-COST IS QUANTITY TIMES UNIT-COST.
```

The REPORTER III program interprets this statement as if you wrote "TOTAL-COST IS QUANTITY * UNIT COST".

B Take this path if the <text> replaces the macro name, unaltered, at every occurrence of the marco name. The same <text> replaces all occurrences of the macro name as report specifications are scanned. The example given for path A illustrates that "*" replaces "TIMES" wherever "TIMES" occurs.

C Take this path to modify the <text> represented by the macro name. The text is modified before it replaces each occurrence of the macro name.

Example:

```
     REPLACE AGED (X) BY AGE (IN WEEKS FROM
        X TO DATE) / 4#.
```

The text associated with "AGED" is altered each time AGED is referenced. For instance, if you write:

```
     REPORT BILLING-AGE IS AGED (INVOICE-DATE).
```

you would get the same effect writing:

```
     REPORT BILLING-AGE IS AGE (IN WEEKS FROM
        INVOICE-DATE TO DATE) / 4.
```

The macro as defined here acts like a new function definition.

D Use formal parameters associated with the macro name to specify how you want to modify <text>. A formal parameter is a word that serves as a place holder for an actual parameter. An actual parameter is the text you enter as a substitute for the formal parameter. Specify actual parameters in report specifications.

At the time the macro name is referenced, the actual parameter (the desired text) replaces the formal parameter. When the <text> replaces the macro name, the supplied actual parameter replaces the formal parameter. You can use the formal parameter one or more times in the body of the text associated with the macro name.

Example:

```
REPLACE TYPE1(N) BY N WHERE TYPE = 1#.
```

When the macro name TYPE1 is referenced, an actual parameter must be supplied for N.

```
REPORT NAME, TYPE1(YEAR), TYPE1(NAME-OF-SUPR).
```

This statement is expanded to:

```
REPORT NAME, YEAR WHERE TYPE = 1, NAME-OF-SUPR
   WHERE TYPE = 1.
```

E Use this path to specify a maximum of ten formal parameters. When the macro name is referenced, the actual parameters are assigned to the formal parameters from left to right. You must specify one actual parameter for each formal parameter.

Example:

```
REPLACE TAX (INC-RATE, CAP-GAIN-R) BY
   SALARY * INC-RATE + CAP-GAIN-R
   * CAPITAL-GAIN#.
```

TAX is then referenced in a report specification as:

```
REPORT INCOME-TAX IS TAX (.15, .0500 * VALUE),
   NAME.
```

In the next step, .15 replaces INC-RATE, and .0500 * VALUE replaces CAP-GAIN-R. The following equivalent statement is the result:

```
REPORT INCOME-TAX IS SALARY * .15 +
.0500 * VALUE * CAPITAL-GAIN, NAME.
```

The same macro could be referenced in a different report specification as:

```
ACCEPT  VALUE-1, VALUE-2.
   SELECT (TAX(VALUE-1, VALUE-2)) > 500.00.
```

In this example, VALUE-1 replaces INC-RATE, and VALUE-2 replaces CAP-GAIN-R. The SELECT statement is expanded to:

```
     SELECT (SALARY * VALUE-1 + VALUE-2 *
        CAPITAL-GAIN) > 500.00.
```

F Take this path after you have specified all formal parameters.

G Take this path to define the <text> that replaces the macro name. The text consists of all characters after the "BY" and before the terminating "#". The text may not contain a "#". The "#" is reserved for indicating the end of the <text>. As the REPLACE statement is scanned, no syntax checking is performed on the text. Syntax checking occurs as the <text> replaces each occurrence of a macro name.

The macro text may contain references to the formal parameter defined in path D above. Actual parameters, which you specify in the report specifications, replace each formal parameter as the <text> replaces the macro name.

The <text> may also contain references to other macro names. These macro name references are not expanded into the associated <text> until the macro name is referenced. Macros may be nested up to 10 deep. The examples for paths D and E illustrate parameters. The following examples illustrate nested macros:

```
     REPLACE A BY B + C #.
     REPLACE B BY D * E #.
     REPLACE C BY F / G #.
```

If you write "REPORT T IS A, X, Y." after entering the above macros, the following replacement steps are taken:

```
     REPORT T IS A, X, Y.
     REPORT T IS B + C, X, Y.
     REPORT T IS D*E + C, X, Y.
     REPORT T IS D*E + F/G, X, Y.
```

Thus, the original statement is equivalent to "REPORT T IS D*E + F/G, X, Y."

CAUTION

Macro names cannot reference themselves. The following specifications result in an infinite series of references and must be avoided.

```
          REPLACE A BY A+B #.
```

This statement is flagged when A is seen in the text of the macro.

```
REPLACE A BY B #.
REPLACE B BY A #.
```

An error is flagged if either A or B is referenced because the limit of 10 nested macros is exceeded.

If a period appears before the "#", problems may arise if, in addition, you use a period after the macro name.

Example:

```
REPLACE ONE BY 1.#.
SELECT COST > ONE.
```

This statement results in SELECT COST > 1.., and causes an error.

H Take this path as many times as necessary to define additional macros in one REPLACE statement. You can also define each macro name in a separate REPLACE statement.

Example:

```
REPLACE PLUS BY + #, INTERVAL(X,Y) BY AGE(IN
    DAYS FROM X TO Y) /30 #.
```

In this case two macros, PLUS and INTERVAL, are defined and added to the vocabulary.

I Take this path after you have completed all macro definitions.

SET OPTION STATEMENT

Use the SET OPTION statement to override default blocking factors for two internal work files generated by the report program, or to force or prevent the building of the vocabulary files with the option DECIMAL-POINT IS COMMA.

You may often want to speed up the report program by allocating more buffer space for these work files. The SET SORT-FILE and SET LOGICAL-PAGE-FILE statements allocate this additional space.

Blocking factor options and DECIMAL-POINT IS COMMA option set in the vocabulary, either by default or with a SET OPTION statement, affect all generated reports using the vocabulary. Use a SET OPTION statement in the report specifications to override the default blocking factor of these files; however, you cannot change the setting of DECIMAL-POINT IS COMMA in the report specifications.

The syntax for the SET OPTION statement is as follows:

```
               A
------>SET------->SORT-FILE--------------->BLOCKING--> (1)
             |                       |
             | B                     |
             |-->LOGICAL-PAGE-FILE-->+
             |
             | C
             |-->DECIMAL-POINT IS COMMA-------------> (2)
             |
             | D
             +-->LANGUAGE---------------------------> (3)
```

```
         E           H
(1) -----------------------> <integer> ----------------->.
       |           |                           |    |
       | F         |                           |    |
       |--->TO---->|                           |    |
       |           |                           |    |
       | G         |                           |    |
       +---> = --->+                           |    |
                                               |    |
         I              L                      |    |
(2) ------------------------------------------->+    |
       |           |  |                 |           |
       | J         |  | M               |           |
       |--->TO---->|  |---->TRUE------->|           |
       |           |  |                 |           |
       | K         |  | N               |           |
       +---> = --->+  +----->FALSE----->+           |
                                                    |
         O              R                           |
(3) -------------------------->COBOL74-------------->+
       |           |  |                 |
       | P         |  | S               |
       |--->TO---->|  +----->COBOL85--->+
       |           |
       | Q         |
       +---> = --->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify the blocking factor of the internal sort file. If you use the report specifications GROUP BY, ORDER BY, or RANGE BY, the internal sort file is used for extracting and sorting information from the data base. Its default blocking factor is 10.
B	Take this path to specify the blocking factor of the logical-page work file. This file stores logical pages for the physical report, and is present only in case of horizontal page overflow. Its default blocking factor is 1.
C	This path is used to force the building of the vocabulary files with the COBOL SPECIAL-NAMES option DECIMAL-POINT IS COMMA if set to TRUE. If set to FALSE, it will prevent the building of the vocabulary files with DECIMAL-POINT IS COMMA.

If this option is not used, source files will be checked for a SPECIAL-NAMES paragraph. If a SPECIAL-NAMES paragraph is found, a check will be made for the DECIMAL-POINT IS COMMA clause. If the clause is found, the vocabulary files will be built with the DECIMAL-POINT IS COMMA just as if the option had been set to TRUE.

D	Take this path to specify the language and the compiler that RP3GEN is to use to create the report program.
E-G	These paths indicate synonyms.
H	Enter a number to set the number of records in a block for the specified work file.
I-K	These paths indicate synonyms.
L-M	These paths indicate synonyms. This forces the building of the vocabulary files with DECIMAL-POINT IS COMMA. This is useful if DECIMAL-POINT IS COMMA is desired and the only input to RP3VOC is DMSII structures or if none of the COBOL source files have a SPECIAL-NAMES paragraph with a DECIMAL-POINT IS COMMA clause.
N	This path prevents the building of the vocabulary files with the option DECIMAL-POINT IS COMMA even though a COBOL source file may have a SPECIAL-NAMES paragraph with the clause DECIMAL- POINT IS COMMA.
O-Q	These paths indicate synonyms.
R	Take this path to specify that RP3GEN is to create a COBOL74 program and use the COBOL74 compiler to compile the program. This is the default setting, that is, you do not have to specify this option to create a COBOL74 report program.
S	Take this path to specify that RP3GEN is to create a COBOL85 program and use the COBOL85 compiler to compile the program. This setting is required if any COBOL source in the vocabulary specifications or any external files accessed by RP3VOC contain COBOL85 source code. Note that COBOL85 can be used only on A Series Systems.

USER NAME STATEMENT

Use the USER NAME statement to enter documentation in the vocabulary. The syntax for the USER NAME statement is as follows:

```
                                        A
---->USER NAME---------------------> <literal> -------> .
          |          |   |                     |
          |-->IS---->|   | B                   |
          |          |   +---> <word> ------->+
          +--> = --->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A-B	You can enter documentation using a <literal> through path A, or using a <word> through path B. The documentation must not exceed 20 alphanumeric characters. A word or literal with more than 20 characters is right-truncated to the first 20 characters. You cannot use character strings of more than 30 characters. Examples: USER NAME IS MIGHTYMOUSE. USER NAME JONES. USER NAME = "JOE DOE".

VOCABULARY STATEMENT

The RP3VOC program generates two files used by the RP3REP program. These two files are used to communicate the data descriptions defined in the vocabulary specification to the report-generation phase of the REPORTER III System. These two files are:

1. The Vocabulary Library file which may contain COBOL source for system file descriptions and/or input procedures.
2. The Vocabulary Dictionary file which contains data names and information related to the data names. When a data name is referenced in a report specification, RP3REP will search the dictionary for the data name and take appropriate action based on the information found.

The VOCABULARY statement must name these two files. There are no defaults. If you do not specify two names in the VOCABULARY statement, the program is terminated. The syntax for the VOCABULARY statement is as follows:

```
---->VOCABULARY-----------------------------------> (1)
                |           |  |           |
                +-->FILES-->+  |-->ARE---->|
                               |           |
                               +--> = ---->+

 (1) --> <external file name> -->AND --> <external file name> --->.
```

The two <external file name>s in the statement are used to identify the vocabulary library and the vocabulary dictionary. The first <external file name> specified is the vocabulary library. This <external file name> is also the collective name for the vocabulary and is the name which is used to identify the vocabulary in a report specification. RP3REP will use this file to find the name of the vocabulary dictionary.

The VOCABULARY statement must be the first statement in a vocabulary specification.

Examples:

```
VOCABULARY FILES ARE "VOCAB1" AND "VOCAB2".
VOCAB FILES = "JLVOC1" AND "JLVOC2".
VOCAB "AVOCAB" AND "BVOCAB".
```

SECTION 4

DMS II LANGUAGE STATEMENTS

This section contains three parts:

1. A Series of Systems DMS II statements.
2. B 1000 Series of Systems DMS II statements.
3. B 2000/B 3000/B 4000 Series of Systems DMS II statements.

Each of these parts begins with an overview of the statements used to add DMS II data-base descriptions to a vocabulary. Following the overview is an example of a vocabulary specification using DMS II, with descriptions of each statement in the specification.

Next, a general discussion and brief descriptions of the data-base information statements associated with DMS II are presented. The remainder of each part lists the statements alphabetically, giving a detailed description and syntax diagram of each statement.

A SERIES OF SYSTEMS
DMS II LANGUAGE STATEMENTS

This section describes how to add one or more A Series DMS II data bases to the vocabulary.

Example:

```
DB UNIV.
END DB.
```

This example adds all elements in the data base named UNIV to the vocabulary. The following is an overview of the syntax (for more detail refer to Data-Base Information):

```
   +<-------------------------------------------------------- (1)
   |
   | A                   B
-------> <DB statement> -----------------------------------> (2)
                          |                               |
                          | C                             |
                          +--> <data-base information> -->+

(1) <---------------------+
                        E |
     D                    | F
(2) ---> <END DB statement> ----->
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <DB statement> identifies an A Series DMS II data base. RP3VOC processes the statements for the data base by referencing a directory file. A utility program, RP3VDM, creates this directory file. RP3VDM must be run once for each physical or logical data base that RP3VOC uses. RP3VOC automatically executes RP3VDM to create the required

directory. You also have the option to run RP3VDM manually. (Refer to Section 7 for details on running RP3VDM).

If the data base is a logical data base, the directory file is called DB/INVOKE/<data-base name>/<logical-data-base name>. If the data base is not a logical data base, the directory is called DB/INVOKE/<data-base name>.

B Take this path to create a default vocabulary for the data base. RP3VOC automatically adds all elements of the data base to the vocabulary.

Default editing pictures are created for all items based on the item attributes specified in DASDL. These editing pictures are used to print items in reports.

The default TOTAL POPULATION of 9999 is assigned to all data sets. TOTAL POPULATION determines default editing pictures used to print out statistical summaries in reports.

C Take this path to specify non-default information on how you want elements of the data base added to the vocabulary. The specification can change names, exclude certain elements, and assign editing pictures and TOTAL-POPULATIONS.

Example:

```
DB CUST-ACCT-INFO.
    DATA SET ACCTS-RECV TOTAL-POP = 200.
    ASSIGN EDITING PICTURE "99/99/99" TO
       DUE-DATE.
    EXCLUDE DATA SET RESTART-AREA.
END DB.
```

This example assigns a TOTAL POPULATION of 200 to ACCTS-RECV, and an editing picture of "99/99/99" to DUE-DATE, but excludes the data set RESTART-AREA and the elements in it. All other elements in the data base CUST-ACCT-INFO are added by default to the vocabulary.

D The END DB statement signals the end of specifications for the data base. All remaining elements in the data base are added by default.

E Take this path to add additional data bases to the vocabulary.

Example:

```
DB UNIV.
END DB.
DB LIBRARY-INFO.
END DB.
```

These statements add all elements of the data bases UNIV and LIBRARY-INFO to the vocabulary.

F Take this path when you have finished specifying all data bases.

VOCABULARY SPECIFICATION EXAMPLE FOR A SERIES DATA BASE

The following example offers you a complete statement-by-statement description of a vocabulary specification for a DMS II data base. The data base UNIV is used in the following specification. Figure 4-1 illustrates the DB-INVOKE listing generated by RP3VDM. This listing represents a DB INVOKE ALL on the data base UNIV. Be sure that you use a current DB-INVOKE listing.

RP3VOC accesses items in the data base in the sequential order displayed in the DB-INVOKE listing, Figure 4-1. Once RP3VOC processes an item, you cannot reference the item again in the vocabulary specification. The following is the vocabulary specification:

```
DB UNIV.
ASSIGN EDITING PICTURE "Z(3)9" TO CRS-NO.
EXCLUDE BOOKS FROM VOCABULARY.
DATASET STUDENTS TOTAL-POP = 300000.
DATASET UNIV-PERSONNEL TOTAL-POP = 500.
SET SS-U-P AS SSNO-PATH.
ASSIGN EDITING PICTURE "$9(4).9(2)" TO SALARY.
REDEFINE UNIV-PERSONNEL AS PROFESSORS
    TOTAL POPULATION = 100.
END DB.
```

The DB UNIV statement causes RP3VOC to initiate several actions. RP3VOC looks for the data-base directory DB/INVOKE/UNIV and readies itself for vocabulary specifications. If DB/INVOKE/UNIV is not present, RP3VDM automatically executes RP3VDM to create this directory for the UNIV data base.

The ASSIGN statement causes RP3VOC to begin processing the elements of the data base sequentially until RP3VOC finds CRS-NO. All the elements preceding CRS-NO are added by default to the vocabulary. You cannot reference any of these items in subsequent vocabulary specifications.

Use the ASSIGN statement to add CRS-NO to the vocabulary with an editing picture of Z(3)9. CRS-NO is printed with this picture in subsequent reports.

The EXCLUDE statement causes RP3VOC to exclude the embedded data set BOOKS, including all its items and sets, from the vocabulary. BOOKS cannot be referenced in report specifications. All data items following CRS-NO and preceding BOOKS are added by default to the vocabulary.

The first DATASET statement causes RP3VOC to add the embedded data set STUDENTS to the vocabulary with an assigned TOTAL POPULATION of 300,000.

The next DATASET statement causes RP3VOC to add the UNIV-PERSONNEL to the vocabulary with an assigned TOTAL POPULATION of 500. A DATASET statement referencing the data item must precede the REDEFINE statement. The remaining data elements of UNIV-COURSES, which are items of the embedded data set STUDENTS, are added to the vocabulary by default.

The SET statement changes the name of the set SS-U-P to SSNO-PATH.

The ASSIGN statement causes RP3VOC to add SALARY to the vocabulary with an editing picture of $9(4).9(2). All elements of UNIV-PERSONNEL preceding SALARY are added by default to the vocabulary.

The REDEFINE statement causes RP3VOC to create another description of the data set UNIV-PERSONNEL under the name PROFESSORS. The data set UNIV-PERSONNEL and its associated elements are duplicated in the data set PROFESSORS. PROFESSORS is assigned a TOTAL POPULATION of 100.

RP3VOC can now reprocess the elements of the data set PROFESSORS. In this example, however, no such references occur. When RP3VOC detects the END DB statement, all elements of the disjoint data set UNIV-PERSONNEL (alias PROFESSORS) are added by default to the vocabulary.

Qualification by the name PROFESSORS or UNIV-PERSONNEL is now required in report specifications to uniquely identify elements within these two data sets.

The END DB statement tells RP3VOC that this is the end of specifications for the data base UNIV. All remaining elements of the DB-INVOKE data-base description are then added to the vocabulary by default.

```
 DB UNIV ALL.
*   02 NOSOFSTUDENTS 9(10) COMP.
*   02 NOSOFCOURSES 9(5) COMP.
*
* 01 UNIV-COURSES DATA SET (#2).
*       UNIV-SET SET (#7,AUTO) OF UNIV-COURSES KEY IS CRS-NAME.
*   02 CNTOFCRS COUNT.
*   02 CRS-NAME.
*       03 DEPARTMENT PIC XX DISPLAY.
*       03 LEVEL PIC 999 COMP.
*       03 CRS-NO PIC 9(4) COMP.
*   02 NOPROF PIC 99 COMP.
*   02 DAYS-OF-WEEK FIELD SIZE IS 06 BITS.
*       03 MON BOOLEAN.
*       03 TUES BOOLEAN.
*       03 WEDS BOOLEAN.
*       03 THURS BOOLEAN.
*       03 FRI BOOLEAN.
*       03 SAT BOOLEAN.
*   02 BUILDING PIC 999 COMP.
*   02 ROOM PIC XX DISLAY.
*   02 COURSENAME PIC X(24) DISPLAY.
*   02 FLAG-BITS FIELD SIZE IS 12 BITS.
*   02 HOURSCRDT PIC 9(4) COMP.
*   02 CLASS-SIZE PIC 99 COMP.
*   02 PROFESSOR REFERENCE TO UNIV-PERSONNEL OCCURS 3.
*   02 BOOKS DATA SET (#3).
*         BOK SET (#4,MANUAL) OF BOOKS KEYDATA IS LC.
*       03 LC PIC 9(9) COMP.
*       03 TITLE PIC X(60) DISPLAY.
*       03 AUTHR PIC X(30) DISPLAY.
*   02 STUDENTS DATA SET (#5).
*         STUDSET SET (#6,AUTO) OF STUDENTS KEYS ARE LAST-NAME,
*         FIRST-NAME.
*       03 LAST-NAME PIC X(15) DISPLAY.
*       03 FIRST-NAME PIC X(10) DISPLAY.
* 01 UNIV-PERSONNEL DATA SET (#8).
*       SS-U-P SET(#9,AUTO) OF UNIV-PERSONNEL KEY IS SSNUM.
*       U-P-SET SET(#10,AUTO) OF UNIV-PERSONNEL KEY IS NAME.
*       FREEPAY SET (#23,AUTO) OF UNIV-PERSONNEL.
*   02 USC-COUNT COUNT.
*   02 NAME.
*       03 LASTNAME PIC X(15) DISPLAY.
*       03 FIRSTNAME PIC X(10) DISLAY.
*   02 SEX BOOLEAN.
*   02 AGE PIC 99 COMP.
*   02 SSNUM PIC 9(9) COMP.
*   02 DPT PIC X(4) DISPLAY.
*   02 RANK PIC X DISPLAY.
*   02 SALARY PIC S9(5)V99 COMP.
*   02 COURSES REFERENCE TO UNIV-COURSES OCCURS 8.
*   02 SUPR REFERENCE TO UNIV-PERSONNEL.
```

Figure 4-1. A Series DB INVOKE LISTING
Of Data Base UNIV

Below is a list of names and associated output editing pictures produced by the vocabulary specification:

UNIV	
NOSOFSTUDENTS	Z(10)
NOSOFCOURSES	Z(5)
UNIV-COURSES	
UNIV-C-SET	
CNTOFCRS	Z(8)
CRS-NAME	
DEPARTMENT	XX
LEVEL	ZZ9
CRS-NO	Z(3)9
NOPROF	Z9
DAYS-OF-WEEK	Z(2)
MON	X(5)
TUES	X(5)
WEDS	X(5)
THURS	X(5)
FRI	X(5)
SAT	X(5)
BUILDING	ZZ9
ROOM	XX
COURSENAME	X(24)
FLAG-BITS	Z(4)
HOURSCRDT	Z(4)
CLASS-SIZE	Z(9)
PROFESSOR	
STUDENTS	
STUDSET	
LAST-NAME	X(15)
FIRST-NAME	X(10)
UNIV-PERSONNEL	
SSNO-PATH	
U-P-SET	
FREEPAY	
USC-COUNT	Z(8)
NAME	
LASTNAME	X(15)
FIRSTNAME	X(10)
SEX	X(5)
AGE	Z9
SSNUM	Z(9)
DPT	X(4)
RANK	X
SALARY	$9(4).9(2)
COURSES	
SUPR	
PROFESSORS	
SS-U-P	
U-P-SET	

FREEPAY	
USC-COUNT	Z(8)
NAME	
LASTNAME	X(15)
FIRSTNAME	X(10)
SEX	X(5)
AGE	Z9
SSNUM	Z(9)
DPT	X(4)
RANK	X
SALARY	-Z(5).99
COURSES	
SUPR	

DATA-BASE INFORMATION

The data-base information statements must access data-base items in the same order as these items occur in the DB-INVOKE listing. The DB-INVOKE listing is a COBOL listing generated by a DB INVOKE ALL. You obtain this listing from running (or having RP3VOC run) RP3VDM. The data-base information statements cause RP3VOC to sequentially process all elements of the DB-INVOKE file until a specific element is referenced. These elements are added to the vocabulary as they are processed by RP3VOC.

All data-base elements are treated sequentially by RP3VOC. Unlike system files, you do not need to add data items individually to the vocabulary. The entire data base or logical data base is added by default; only data sets, sets, and data items that you specify are modified or excluded from the vocabulary. The following is a list of data-base information statements and their descriptions:

Statement	Explanation
ASSIGN Statement	Use the ASSIGN statement to specify an editing picture for a data item. The editing picture is used to print the data item in subsequent reports. If you do not specify an editing picture for a data item, RP3VOC automatically generates a default editing picture from the description of the item given in DASDL. Since the default editing picture may not provide the best form for printing, you can specify a new editing picture to override the default generated by RP3VOC.
DATASET Statement	The DATASET statement specifies a data set. You can use this statement three ways: 1. To assign a TOTAL POPULATION to a data set. The REPORTER III system uses TOTAL POPULATION to determine the number of spaces required for printing statistical summaries. Examples of statistical summaries are TOTAL and COUNT report specifications. 2. To give an alternate name to a disjoint data set.

3. To allow you to redescribe a disjoint data set using the REDEFINE statement. You must add a disjoint data set with the DATASET statement before you use the REDEFINE statement.

EXCLUDE Statement

Use the EXCLUDE statement to delete data sets and associated elements from the vocabulary. Use this statement to exclude sensitive or irrelevant items. Elements excluded from the vocabulary cannot be reported or referenced in report specifications.

REDEFINE Statement

Use the REDEFINE statement to reenter a disjoint data set into the vocabulary under a different name. The REDEFINE statement allows you to create a duplicate of the disjoint set. You can then redescribe elements of the disjoint set of the vocabulary while the original data-set descriptions remain under the original name. You must previously add the disjoint set to the vocabulary using the DATASET statement.

SET Statement

Use the SET statement to give an alternate name to a disjoint set.

ASSIGN Statement

Use the ASSIGN statement to specify an editing picture for a data item. This picture is used to print the data item in reports.

If you do not specify an editing picture for an elementary data item, RP3VOC generates a default editing picture from the description of the data item in the DB-INVOKE listing. The syntax for the ASSIGN statement is as follows:

```
----->ASSIGN------------------->PICTURE-----------------> (1)
           |                 |
           +--->EDITING--->+

                                A
(1) --> "<COBOL picture>" -->TO--> <DMS II item name> --> (2)

         B
(2) -----------------------------> .
      |                          |
      | C                        |
      +--> <qualification> -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <DMS II item name> must be an unsubscripted elementary DMS II data item. For example, assume a data item, DATE-NEEDED, with a NUMBER (6) description is used to store a date. The default editing picture is Z(6). If the printed image of DATE-NEEDED is to be MM/DD/YY, then the required result is obtained in the report by issuing the following statement to RP3VOC: ASSIGN PIC "99/99/99" TO DATE-NEEDED.

Examples:

```
ASSIGN EDITING PICTURE "$Z(9).9(2)" TO
   SALARY.
ASSIGN PIC "999" TO AGE.
```

CAUTION

The COBOL editing picture cannot have any leading or trailing blanks. If the editing picture is smaller than the storage picture, truncation may result.

If you have set the option DECIMAL-POINT IS COMMA, then the appropriate picture must be given to avoid misalignment or COBOL syntax errors.

Examples:

If the option DECIMAL-POINT IS COMMA is not set,

```
ASSIGN PIC "ZZZ,ZZZ.99" TO QTY.
```

If the option DECIMAL-POINT IS COMMA is set,

```
ASSIGN PIC "ZZZ.ZZZ,99" TO QTY.
```

B Take this path if you do not require <qualification>.

C Take this path if you require <qualification> to specify the <DMS II item name>.

For a detailed description of <qualification>, see Section 2.

DATASET Statement

Use the DATASET statement to give an alternate name to a disjoint data set, or to assign a TOTAL POPULATION to a data set.

The TOTAL POPULATION is used to determine how much space to allow for the printing of statistical summaries, such as TOTAL and VARIANCE. It is used to set default editing attributes for certain statistical items.

This statement also allows you to redescribe a disjoint data set using a REDEFINE statement. You must add a disjoint data set to the vocabulary with a DATASET statement before you use a REDEFINE statement. The syntax for the DATASET statement is as follows:

```
            A                         B
--->DATASET---> <data-set name> ----------------------------------> (1)
                                    |                       |
                                    | C                     |
                                    +--> <qualification> -->+


          D                     F
(1) --------------------------------------------------------------> (2)
        |                   | |
        | E                 | | G                   H
        +-->AS--> <name> -->+ +-->TOTAL-POPULATION---------------> (3)
                                                     |        |
                                                     | I      |
                                                     |-->IS-->|
                                                     |        |
                                                     | J      |
                                                     +--> = ->+


(2) -----------------------> .
                       |
      K                |
(3) ----> <integer> -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <data-set name> must be a valid A Series DMS II data set in the DB-INVOKE listing.
B	Take this path if you do not require <qualification> to specify the data set.
C	Take this path if you require <qualification> to specify the data set. For a detailed description of <qualification>, see Section 2.
D	Take this path if you do not use an alternate name to identify the current description of the data set in the vocabulary. For an embedded data set, you must take this path. Example: `DATASET UNIV-COURSES.`
E	Take this path only if you are using an alternate name to identify a disjoint data set in the vocabulary description of the data set. The <name> you specify becomes the internal name for the data set in the generated report program. If you take this path, the <data-set name> in path A no longer exists in the vocabulary, and you cannot reference that name in report specifications. Example: `DATASET UNIV-PERSONNEL AS PROFESSORS.`
F	Take this path if you use the default TOTAL POPULATION of 9999.

G Take this path to override the default TOTAL POPULATION.

Examples:

```
DATASET BOOKS TOTAL-POP IS 300000.
DATASET UNIV-PERSONNEL AS PROFESSORS
TOTAL POPULATION = 997.
```

H-J These paths are equivalent.

K The <integer> specifies the TOTAL POPULATION.

DB Statement

Use the DB statement to instruct RP3VOC that the specifications refer to a specific A Series DMS II data base or logical data base. RP3VOC assumes that all statements in the vocabulary are descriptions of items in the specified data base until RP3VOC encounters an END DB statement. The syntax for the DB statement is as follows:

```
          A
-->DB-----> <data-base name> --------------------------------> (1)
       |                                                 |
       | B                                               |
       +---> <logical-data-base name> OF <data-base name>->+


           C
(1) ------------------------------> .
         |                    |
         | D                  |
         +--->AS---> <name> -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if the data base is not a logical data base. The <data-base name> must be a valid A Series DMS II <data-base name>. The DASDL description file, DESCRIPTION/<data-base name>, or the data-base directory DB/INVOKE/<data-base name> must be present during the execution of RP3VOC. If the data-base directory file is not present, RP3VOC automatically executes RP3VDM to obtain a directory file from the description file. Example: DB UNIV.
B	Take this path if the data base is a logical data base. The <logical-data-base name> must be a valid A Series DMS II <logical-data-base name> that was declared in DASDL. When a logical data base is specified, RP3VOC adds it to the vocabulary information.

The <data-base name> must be a valid A Series DMS II <data-base name>. The DASDL description file, DESCRIPTION/<data-base name>, or the data-base directory, DB/INVOKE/<data- base name>/<logical-data-base name> must be present during the execution of RP3VOC. If the data-base directory file is not present, RP3VOC automatically executes program RP3VDM to obtain the directory file.

Example:

```
DB LOG-UNIV1 OF UNIV.
```

C Take this path if you use the actual data base name rather than an internal (or alias) name. The name of the data base in the vocabulary will be either the actual <data-base name> if you choose path B, or the <logical-data-base name> if you choose path C.

D Take this path if you use an internal (or alias) name for the data base. The name of the data base in the vocabulary will be the <name> you specify here. You can add a data base to a vocabulary a number of times by using alias names.

Examples:

```
DB UNIV AS ALIAS-UNIV.
DB LOG-UNIV OF UNIV AS ALIAS-LOG-UNIV.
```

<u>END</u> <u>DB</u> <u>Statement</u>

The END DB statement instructs RP3VOC to add the remaining elements of the DB-INVOKE description to the vocabulary. Each DB statement must have a corresponding END DB statement. The syntax for the END DB statement is as follows:

```
---------->END DB----------->.
```

Example:

```
     END DB.
```

EXCLUDE Statement

Use the EXCLUDE statement to delete data sets and associated elements from the vocabulary. By default, all elements in the data base are added to the vocabulary unless they are explicitly excluded. Use this statement to exclude sensitive or irrelevant information. The syntax for the EXCLUDE statement is as follows:

```
                A                I
-->EXCLUDE----------------------> <DMS II name> ---> (1)
              |                 |
              | B               |
              |-->DATA SET----->|
              |                 |
              | C               |
              |-->SET---------->|
              |                 |
              | D               |
              |-->SUBSET------->|
              |                 |
              | E               |
              |-->LINK ITEM---->|
              |                 |
              | F               |
              |-->CONTROL ITEM->|
              |                 |
              | G               |
              |-->DATA ITEM---->|
              |                 |
              | H               |
              +-->GROUP ITEM--->+

        J
(1) ---------------------------------------------> .
      |                    |  |                 |
      | K                  |  +-->FROM VOCABULARY-->+
      +---> <qualification> -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if you do not document the type of excluded element.

Example:

```
     EXCLUDE SALARY.
```

B-H Take these paths to document the type of the element you are excluding (for documentation purposes only).

Example:

```
     EXCLUDE DATA SET UNIV-COURSES.
```

I Identify a data base element by a <DMS II name>. This name must reference a valid A Series DMS II element described in the DB-INVOKE listing. Once you exclude a data base element, you cannot reference the element in report specifications.

If a <DMS II name> is a data set, all its elements, including any embedded data sets, are deleted from vocabulary.

If <DMS II name> is a group item, all subordinate elements are deleted from the vocabulary.

If <DMS II name> is a set or subset, it is marked as excluded in the vocabulary file.

J Take this path if you do not require <qualification> to specify the <DMS II name>.

K Take this path if you require <qualification> to specify the <DMS II name>.

For a detailed description of <qualification>, see Section 2.

REDEFINE Statement

Use the REDEFINE statement to reenter a disjoint data set into the vocabulary under an alias name. With the REDEFINE statement, you can create another description of a data set under a different name. You can change or exclude any item in the data set by following the REDEFINE statement with the necessary ASSIGN, EXCLUDE, or SET statement.

You can also have multiple REDEFINE statements referencing the same data set, creating several different descriptions of the data set under different names.

To use the REDEFINE statement, you must have previously added the data set to the vocabulary via a DATASET statement. No other DATASET statements referencing additional disjoint data sets can be made before the REDEFINE statement for that particular data set. The syntax for the REDEFINE statement is as follows:

```
                                    A
--->REDEFINE-------------------> <data-set name> ----> (1)
             |               |
             +--->DATASET--->+


                     B
(1) --->AS----------> <name> ----------------------> (2)


       C
(2) ---------------------------------------------------------> .
     |                                                    |
     | D                     E        H                   |
     +--->TOTAL-POPULATION---------------> <integer> --->+
                             |        |
                             | F      |
                             |--->IS--->|
                             |        |
                             | G      |
                             +---> = --->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <data-set name> identifying the data set must be a valid A Series DMS II disjoint data set described in the DB-INVOKE listing.
B	The <name> becomes the alias name of the duplicate data set.
C	Take this path if you use the default TOTAL POPULATION of 9999. The REPORTER III system uses TOTAL POPULATION to determine editing pictures when reporting statistics on items in the redefined data set.
D	Take this path to override the default TOTAL POPULATION.
E-G	These paths are equivalent.
H	The <integer> specifies the TOTAL POPULATION.

Example:

```
        .
        .
        .
   DATASET UNIV-PERSONNEL TOTAL-POP = 997.
   ASSIGN PIC "$ZZ,ZZ9.99" TO SALARY.
   REDEFINE UNIV-PERSONNEL AS UNIV-SUPR
      TOTAL-POP = 200.
   EXCLUDE SALARY.
        .
        .
        .
```

The above example allows the following INPUT statement in the report specifications:

```
   INPUT UNIV-PERSONNEL, UNIV-SUPR VIA SUPR.
```

SET Statement

Use the SET statement to change the name of a particular disjoint set. Only use a SET statement to resolve a possible conflict in names. The <set name> must be a valid DMS II disjoint set described in the DB-INVOKE listing. The syntax for the SET statement is as follows:

```
                                      A
—>SET—> <set name> ———————————————————————————————————> (1)
                                    |                        |
                                    | B                      |
                                    +—-> <qualification> —->+


     C
(1) ————>AS———> <name> ————————————————————————————————> .
```

The paths of this syntax diagram are explained below.

PATH	Explanation
A	Take this path if you do not require <qualification> to specify the disjoint set.
B	Take this path if you require <qualification> to specify the disjoint set. For a detailed description of <qualification>, see Section 2.
C	The <name> becomes the internal name of the set. You can no longer use the <set name> to reference the disjoint data set in the report specifications. Example: SET SS-U-P AS SSN-SET.

B 1000 SERIES OF SYSTEMS
DMS II LANGUAGE STATEMENTS

This section describes how to add one or more B 1000 Series DMS II data bases to the vocabulary.

Example:

```
DB UNIV.
END DB.
```

This example adds all elements in the data base UNIV to the vocabulary. The following is an overview of the syntax. (For more detail, refer to Data-Base Information.)

```
  +<-------------------------------------------------------- (1)
  |
  | A                     B
------> <DB statement> ---------------------------------> (2)
                        |                              |
                        | C                            |
                        +--> <data-base information> ->+

(1) <----------------------+
                          E|
    D                      | F
(2) ---> <END DB statement> ------>
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <DB statement> informs RP3VOC that the specifications refer to a B 1000 Series data base. RP3VOC processes B 1000 Series statements by accessing data-set COBOL library files created by DASDL. The format for the library file name is #<data-base name>/<disjoint-data-set name>.

You must specify B 1000 Series DMS II data sets individually to RP3VOC. You may specify these data sets in any order. The following data-set items are added by default to the vocabulary:

1. Sets.
2. Subsets.
3. Group items.
4. Data items.

You must specify disjoint data sets individually, and in the sequential order that disjoint data sets appear in the following listings:

1. A DMPALL listing of the COBOL library file.

2. A COBOL compiler listing containing an INVOKE statement for the data set.

3. A listing obtained by using the "$SOURCE" option when compiling with DASDL.

B Take this path to specify how you want data sets and disjoint data sets added to the vocabulary.

Example:

```
DB UNIV.
DATASET UNIV-PERSONNEL.
END DB.
```

These statements add only the disjoint data set UNIV-PERSONNEL of the data base UNIV to the vocabulary. All elements of UNIV-PERSONNEL are added by default. No other disjoint data sets are added to the vocabulary. The <END DB statement> signals the end of specifications for the data base.

C Take this path to add additional data bases to the vocabulary.

Example:

```
DB UNIV.
  :
END DB.
DB LIBRARY-INFO.
  :
END DB.
```

This example adds the data bases UNIV and LIBRARY-INFO to the vocabulary.

D Take this path when you have finished specifying all data bases.

<u>RP3VOC SPECIFICATION EXAMPLE FOR B 1000 SERIES DATA BASE</u>

The following example gives you a complete statement-by-statement description of a vocabulary specification for a DMS II data base. The data base UNIV is used in the following vocabulary specification. Figure 4-2 is an example of the DASDL description of the data base UNIV. RP3VOC accesses items in the data base in the sequential order displayed in the INVOKE listing. Figure 4-3 illustrates the INVOKE listing that you generate by:

1. A DMPALL listing the data-set library files.
2. Compiling DASDL with the $SOURCE option.
3. Specifying "INVOKE" on the data sets in a COBOL compilation.

The following is the vocabulary specification:

```
DB UNIV.
DATASET COURSES.
ASSIGN EDITING PICTURE "Z(3)9" TO CRS-NO.
EXCLUDE BOOKS FROM VOCABULARY.
DATASET PERSONNEL TOTAL-POP= 999
ASSIGN EDITING PICTURE "9(4).9(2)" TO
  SALARY.
REDEFINE PERSONNEL AS PROFESSORS
  TOTAL POPULATION 999.
END DB.
```

Each statement in this example produces the following results:

The DB UNIV statement causes RP3VOC to ready itself for upcoming UNIV data base specifications.

The first DATASET statement causes RP3VOC to open the COBOL library file #UNIV/COURSES and make an entry in the vocabulary for this disjoint data set.

The first ASSIGN statement causes RP3VOC to begin sequentially processing the elements of the data set until RP3VOC finds CRS-NO. All the elements preceding CRS-NO are added by default to the vocabulary, along with the RP3VOC-supplied default editing pictures. You cannot reference any of these items in subsequent vocabulary specifications.

Use the ASSIGN statement to add CRS-NO to the vocabulary with an editing picture of Z(3)9. CRS-NO is printed with this picture in reports.

The EXCLUDE statement causes RP3VOC to exclude the embedded data set BOOKS, including all its elements, from the vocabulary. BOOKS cannot be referenced in report specifications. All data items following CRS-NO and preceding BOOKS are added to the vocabulary by default.

The next DATASET statement causes RP3VOC to add PERSONNEL to the vocabulary with an assigned TOTAL POPULATION of 999. Since RP3VOC cannot find PERSONNEL as an embedded data set of COURSES, RP3VOC accesses the COBOL library file #UNIV/PERSONNEL. The remaining elements of COURSES, which are part of the subset STUDENTS, are added by default to the vocabulary. The DATASET statement referencing the data set must precede the REDEFINE statement.

The next ASSIGN statement causes RP3VOC to add SALARY to the vocabulary with an editing picture of $9(4).9(2). All elements of PERSONNEL preceding SALARY are added by default to the vocabulary.

The REDEFINE statement causes RP3VOC to create another description of the data set PERSONNEL under the name PROFESSORS. PERSONNEL and its associated data elements are duplicated in the data set PROFESSORS. PROFESSORS is assigned a TOTAL POPULATION of 999.

RP3VOC can now reprocess the elements of the data set PROFESSORS. In this example, however, no such references occur. When RP3VOC detects the END DB statement, all elements of the disjoint data set PERSONNEL (alias PROFESSORS) are added by default to the vocabulary.

Qualification by the name PROFESSORS or PERSONNEL is now required in report specifications to uniquely identify elements within these two data sets.

The END DB statement tells RP3VOC that this is the end of the data base specifications.

```
:%THIS DASDL PROGRAM GIVES EXAMPLES
:%OF THE VARIOUS CONSTRUCTS USED IN
:%DASDL TO DESCRIBE A DATA BASE.
:PARAMETERS(
:    BUFFERS = 10  );
:COURSES DATA SET "MAIN FILE"(
:    CRS-NAME GROUP (
:         DEPARTMENT ALPHA (2);
:         LEVEL NUMBER(3);
:         CRS-NO NUMBER (4));
:    NOPROF NUMBER (2);
:    DAYS-OF-WEEK GROUP (
:         MON NUMBER(1);
:         TUES NUMBER(1);
:         WEDS NUMBER(1);
:         THURS NUMBER(1);
:         FRI NUMBER(1);
:         SAT NUMBER(1));
:     BUILDING NUMBER(3);
:     ROOMNUMBER ALPHA(2);
:     COURSENAME ALPHA(24);
:     FLAG-BITS ALPHA(12);
:     HOURSCRDT NUMBER(4);
:     CLASS-SIZE NUMBER(2);
:     PROFESSOR SUBSET OF PERSONNEL, POPULATION = 3;
:     BOOKS UNORDERED DATA SET(
:         LC NUMBER(9);
:         TITLES ALPHA(60);
:         AUTHR ALPHA(30);
:     STUDENTS SUBSET OF MSF KEY IS
:         (LNAME,FNAME)DUPLICATES,
:         POPULATION = 300
:     POPULATION = 1000;
:     UNIV-C-SET ORDERED SET OF COURSES
:                KEY IS (CRS-NAME)
:PERSONNEL DATASET(
:     NAME GROUP(
:         LASTNAME ALPHA(15);
:         FIRSTNAME ALPHA(10);
:     SEX NUMBER(1);
:     AGE NUMBER(2);
:     SSNUM NUMBER(9);
:     DPT ALPHA(4);
:     RANK ALPHA(1);
:     SALARY NUMBER(S7.2);
:     COURSES SUBSET OF COURSES, POPULATION = 8;
:     ADDRES SUBSET OF ADR;
:     SUPR SUBSET OF PERSONNEL);
:     SS-U-P ORDERED SET OF PERSONNEL KEY IS (SSNUM);
:     U-P-SET ORDERED SET OF PERSONNEL KEY IS
:           (LASTNAME,FIRSTNAME) DUPLICATES;
:MSF DATA SET(
:    SSNO NUMBER(9);
:    NONAM NUMBER(1);
:    LNAME ALPHA(30);
:    MNAME ALPHA(30);
:    FNAME ALPHA(30);
```

Figure 4-2. Example of B 1000 Series Data and Structure Definition Language (DASDL) (Sheet 1 of 2)

```
:    CAMPUS-ADDRESS GROUP(
:        DORM ALPHA(6);
:        ROOM NUMBER(4);
:        POBOX NUMBER(4);
:        PHONE NUMBER(7);
:    ND NUMBER(2);
:    DEGREE ALPHA(4) OCCURS 6 TIMES;
:    TOTHRS NUMBER(3);
:    TOTOP NUMBER(3);
:    GRADE-POINT-AVG NUMBER (3.2);
:    MJR NUMBER(3);
:    AMJR ALPHA(18);
:    SSEX NUMBER(1);
:    SAGE NUMBER(2);
:    HOME-ADDRESS SUBSET OF ADR;
:    QUARTER   ORDERED DATA SET(
:        QTR ALPHA(4);
:        QTTHRS NUMBER(2);
:        QTROP NUMBER(2);
:        CORSES ORDERED DATA SET(
:            TYPECOURSE NUMBER(1);
:            YR NUMBER(2);
:            Q NUMBER(2);
:            GCRS SUBSET OF UNIV-COURSES;
:            GGD ALPHA(2);
:            TITLE-OF-PAPER ALPHA(30);
:            PPRGD ALPHA(2);
:            POPULATION = 4;
:            CSET ACCESS TO CORSES KEY IS
:                 (TYPECOURSE) DUPLICATES)
:        POPULATION = 5000;
:        QSET ACCESS TO QUARTER KEY IS (QTR));
:    MSFSET ORDERED SET OF MSF KEY IS (SSNO);
:ADR DATA SET(
:    FACULTY-STUDENT NUMBER(1);
:    SNO NUMBER(9);
:    ADLN ALPHA(54) OCCURS 9 TIMES;
:    ZIPC NUMBER(5);
:    PHON NUMBER(10));
:    SSAD ORDERED SET OF ADR KEY IS (SNO);
:BOOKS(
:    AREASIZE  500,
:    TYPE = UNORDERED LIST,
:    BLOCKSIZE = 5);
:BOOKFILE STORAGE FOR BOOKS(
:    TITLE = UNIV/LIBRARY,
:    AREAS = 10);
:UNIV-C-SET(
:    TABLESIZE = 12,
:    AREASIZE = 10,
:    TYPE = INDEX SEQUENTIAL.
:    LOADFACTOR = 9);
:PERSONNEL(
:    PRIME,
:    POPULATION = 997);
:INITIALIZE;
:$FILE STRUCTURE
```

Figure 4-2. Example of B 1000 Series Data and Structure Definition Language (DASDL) (Sheet 2 of 2)

```
01   COURSES DATA SET.....
     UNIV-C-SET ORDERED SET.....OF COURSES.....
             KEY IS CRS-NO.
     02   CRS-NAME.
       03   DEPARTMENT                               PIC X(2).
       03   LEVEL                                    PIC 9(3) COMP.
       03   CRS-NO                                   PIC 9(4) COMP.
     02   NOPROF                                     PIC 9(2) COMP.
     02   DAYS-OF-WEEK.
       03   MON                                      PIC 9 COMP.
       03   TUES                                     PIC 9 COMP.
       03   WEDS                                     PIC 9 COMP.
       03   THURS                                    PIC 9 COMP.
       03   FRI                                      PIC 9 COMP.
       03   SAT                                      PIC 9 COMP.
     02   BUILDING                                   PIC 9(3) COMP.
     02   ROOMNUMBER                                 PIC X(2).
     02   COURSENAME                                 PIC X(24).
     02   FLAG-BITS                                  PIC X(12).
     02   HOURSCRDT                                  PIC 9(4) COMP.
     02   CLASS-SIZE                                 PIC 9(2) COMP.
     02   PROFESSOR SUBSET.....
     02   BOOKS DATASET....
       03   LC                                       PIC 9 COMP.
       03   TITLES                                   PIC X(60).
       03   AUTHR                                    PIC X(30).
     02   STUDENTS SUBSET.....

01   PERSONNEL DATA SET.....
     SS-U-P ORDERED SET.....
     U-P-SET ORDERED SET
     02   NAME.
       03   LASTNAME                                 PIC X(15).
       03   FIRSTNAME                                PIC X(10).
     02   SEX                                        PIC 9 COMP.
     02   AGE                                        PIC 9(2) COMP.
     02   SSNUM                                      PIC 9(9) COMP.
     02   DPT                                        PIC X(4).
     02   RANK                                       PIC X.
     02   SALARY                                     PIC S9(6)V99 COMP.
     02   COURSES SUBSET.....
     02   ADDRES SUBSET....
     02   SUPR SUBSET....
```

Figure 4-3. B 1000 Series Listing Of Disjoint Data Sets

The following is a list of the vocabulary of names and associated editing pictures produced by the vocabulary specification:

UNIV	
COURSES	
UNIV-C-SET	
CRS-NAME	
DEPARTMENT	XX
LEVEL	ZZ9
CRS-NO-Z	Z(3)9
NOPROF	Z9
DAYS-OF-WEEK	
MON	9
TUES	9
WEDS	9
THURS	9
FRI	9
SAT	9
BUILDING	ZZ9
COURSENAME	X(24)
FLAG-BITS	X(12)
HOURSCRDT	Z(3)9
CLASS-SIZE	Z9
PROFESSOR	
STUDENTS	
PERSONNEL	
SS-U-P	
U-P-SET	
NAME	
LASTNAME	X(15)
FIRSTNAME	X(10)
SEX	9
AGE	Z9
SSNUM	Z(8)9
DPT	X(4)
RANK	X
SALARY	$9(4).9(2)
COURSES	
SUPR	
PROFESSORS	
SS-U-P	
NAME	
LASTNAME	X(15)
FIRSTNAME	X(10)
SEX	9
AGE	Z9

SSNUM	Z(8)9
DPT	X(4)
RANK	X
SALARY	-Z(5).99
COURSES	
SUPR	

DATA-BASE INFORMATION

Both data sets and disjoint data sets must be specified individually to be added to the vocabulary. Embedded data sets and other data-set items are added by default once the data set is specified.

If you want to exclude or change any elements in a disjoint data set, you must present these elements in the sequential order displayed in a DMPALL listing. Any element not specifically referenced is added by default to the vocabulary with default editing pictures. The following is a list of data-base information statements and their descriptions:

Statement	Explanation
ASSIGN Statement	Use the ASSIGN statement to specify an editing picture for a data item. The editing picture is used to print the data item in subsequent reports. If you do not specify an editing picture for a data item, RP3VOC automatically generates a default editing picture from the description of the item given in DASDL. Since the default editing picture may not provide the best form for printing, you can specify a new editing picture to override the default generated by RP3VOC.
DATASET Statement	The DATASET statement specifies a data set. You can use this statement three ways: 1. To assign a TOTAL POPULATION to a data set. The REPORTER III system uses TOTAL POPULATION to determine the number of spaces required for printing statistical summaries. Examples of statistical summaries are TOTAL and COUNT report specifications.

2. To give an alternate name to a disjoint data set.

3. To allow you to redescribe a disjoint data set using the REDEFINE statement. You must add a disjoint data set with the DATASET statement before you use the REDEFINE statement.

EXCLUDE Statement

Use the EXCLUDE statement to delete data sets and associated sensitive or irrelevant items. Elements excluded from the vocabulary cannot be reported or referenced in report specifications.

REDEFINE Statement

Use the REDEFINE statement to reenter a disjoint data set into the vocabulary under a different name. The REDEFINE statement allows you to create a duplicate of the disjoint set. You can then redescribe elements of the disjoint set of the vocabulary while the original data-set descriptions remain under the original name. You must previously add the disjoint set to the vocabulary using the DATASET statement.

ASSIGN Statement

Use the ASSIGN statement to specify an editing picture for a data item. This picture is used to print the data item in reports.

If you do not specify an editing picture for an elementary data item, RP3VOC generates a default editing picture from the description of the data item in the COBOL library file. The syntax for the ASSIGN statement is as follows:

```
------>ASSIGN------------------->PICTURE------------------> (1)
            |                |
            +--->EDITING--->+


                                   A
(1) ---> "<COBOL picture>" --->TO----> <DMS II item name> -> (2)


        B
(2) ------------------------------> .
       |                           |
       | C                         |
       +----> <qualification> ->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <DMS II item name> must be an unsubscripted elementary DMS II data item.

For example, assume that a data item, DATE-NEEDED, with a NUMBER (6) description is used to store a date. The default editing picture is Z(6). If the printed image of DATE-NEEDED is to be MM/DD/YY, then you can obtain the result required in this report by issuing the following statement to RP3VOC:

```
ASSIGN PIC "99/99/99" TO DATE-NEEDED.
```

Examples:

```
ASSIGN EDITING PICTURE "$Z(9).9(2)"
  TO SALARY.
ASSIGN PIC "999" TO AGE.
```

CAUTION

The COBOL editing picture cannot have any leading or trailing blanks. If the editing picture is smaller than the storage picture, truncation may result.

If you have set the option DECIMAL-POINT IS COMMA, then the appropriate picture must be given to avoid misalignment or COBOL syntax errors.

Examples:

If the option DECIMAL-POINT IS COMMA is not set,

```
ASSIGN PIC "ZZZ,ZZZ.99" TO QTY.
```

If the option DECIMAL-POINT IS COMMA is set,

```
ASSIGN PIC "ZZZ.ZZZ,99" TO QTY.
```

B Take this path if you do not require <qualification>.

C Take this path if you require <qualification> to specify the <DMS II item name>.

For a detailed description of <qualification>, see Section 2.

DATASET Statement

The DATASET statement adds a data set to the vocabulary. Use the DATASET statement to give an alternate name to a disjoint data set, or to assign a TOTAL POPULATION to a data set. The TOTAL POPULATION is used to determine how much space to allow for the printing of statistical summaries, such as TOTAL and VARIANCE. TOTAL POPULATION is also used to set default editing attributes for certain statistical items.

This statement further allows you to redescribe a disjoint data set using a REDEFINE statement. You must use a DATASET statement to add a disjoint data set to the vocabulary before you use a REDEFINE statement to redescribe it. The syntax for the DATASET statement is as follows:

```
                A
-->DATASET-----> <data-set name> -----------------------> (1)


          B                          D
(1) ---------------------------------------------------> (2)
          |                        | |                 |
          | C                      | | E               |
          +---> <qualification> ->+ +--->AS--> <name> ->+


          F
(2) ------------------------------------------------------> .
          |                                               |
          | G                  H        K                 |
          +--->TOTAL-POPULATION------------> <integer> ->+
                              |       |
                              | I     |
                              |-->IS-->|
                              |       |
                              | J     |
                              +--> = ->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <data-set name> must be a valid B 1000 Series DMS II data set.

RP3VOC first looks for an embedded data set called <data-set name>. If the <data-set name> is not found in the current COBOL library file, RP3VOC attempts to open and process the COBOL library file, #<data-base name>/<data-set name>. RP3VOC then looks for a disjoint data set called <data-set name>.

If one of the following conditions is present:

1. The COBOL library file #<data-base name>/<data-set name> is not on disk, or

2. The <data-base name>, <logical-data-base name>, or <data-set name> is misspelled,

then RP3VOC will wait with a NO FILE message from the MCP until the file is created or present, or until a SPO IL message is entered.

B Take this path if you do not require <qualification> to specify the data set.

C Take this path if you require <qualification> to specify the data set.

For a detailed description of <qualification>, see Section 2.

D Take this path if you do not use an alternate name to identify the current description of the data set in the vocabulary. For an embedded data set, you must take this path.

Example:

```
DATASET PERSONNEL.
```

E Take this path only if you are using an alternate name to identify a disjoint data set in the vocabulary description of the data set. The <name> you specify becomes the internal name for the data set in the generated report program. If you take this path, the <data-set name> in path A no longer exists in the vocabulary, and you cannot reference that name in report specifications.

Example:

DATASET PERSONNEL <u>AS</u> <u>PROFESSORS.</u>

F Take this path if you use the default TOTAL POPULATION of 9999. For an embedded data set, the TOTAL POPULATION reflects the total number of members in the embedded data set for all members of the owner data set.

G Take this path to override the default TOTAL POPULATION.

Examples:

```
DATASET BOOKS TOTAL-POP IS 1500.
DATASET PERSONNEL AS PEOPLE
   TOTAL-POP 100.
```

H-J These paths are equivalent.

K The <integer> specifies the TOTAL POPULATION.

DB Statement

Use the DB statement to instruct RP3VOC that the specifications refer to a B 1000 Series DMS II data base. RP3VOC assumes that all statements in the vocabulary are descriptions of items in the specified data base until RP3VOC encounters an END DB statement. The syntax for the DB statement is as follows:

```
          A
->DB------------------------------------------------>(1)
      |                                        |
      | B                                      |
      +---> <logical-data-base name>->OF-->+


                            C
(1) -> <data-base name> ----------------------------->
                           |                      |
                           | D                    |
                           +--->ON "<pack ID>" -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if the data base is not a logical data base, and logical data bases will not be included. The <data-base name> must be a valid B 1000 Series <data-base name> declared in DASDL. If you want to add disjoint data sets, the COBOL library files must be present.
B	Take this path if the data base is a logical data base. The <logical-data-base name> must be a valid B 1000 Series DMS II <logical-data-base name> declared in DASDL. If you want to add disjoint data sets, the COBOL library files must be present. Example: DB LOG-UNIV OF UNIV. RP3VOC cannot check the validity of the physical <data-base name>, nor can it check whether the logical data base belongs to the physical data base. You must specify the LIST GO option to check the physical or logical data base (see LIST Statement).

C Take this path if you do not declare a <pack ID> (i.e., the dictionary file and all of the COBOL library files are on disk).

D Take this path if the COBOL library files and the dictionary file of the data base reside on pack. The <pack ID> is a string with a maximum of ten characters. The COBOL library files and the dictionary file of a data base must be on the same storage device (i.e., disk or pack) when executing RP3VOC.

Example:

```
DB LOG-UNIV OF UNIV ON "USERPACK".
```

END DB Statement

The END DB statement instructs RP3VOC to add the remaining elements of the current data-set COBOL library file to the vocabulary. Each DB statement must have a corresponding END DB statement. The syntax for the END DB statement is as follows:

```
--------->END DB------------------->  .
```

Example:

```
     END DB.
```

EXCLUDE Statement

Use the EXCLUDE statement to exclude elements in a disjoint data set from the vocabulary. By default, all elements in the data base are added to the vocabulary unless they are explicitly excluded. Use this statement to exclude sensitive and irrelevant information. The syntax for the EXCLUDE statement is as follows:

```
             A                G
--->EXCLUDE---------------------> <DMS II name> --> (1)
           |                |
           | B              |
           |--->DATA SET--->|
           |                |
           | C              |
           |--->SET-------->|
           |                |
           | D              |
           |--->SUBSET----->|
           |                |
           | E              |
           |--->DATA ITEM-->|
           |                |
           | F              |
           +--->GROUP ITEM->+

        H
(1) ----------------------------------------------> (2)
       |                     |
       | I                   |
       +---> <qualification> ->+

(2) ----------------------------------------------> .
       |                   |
       +-->FROM VOCABULARY-->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if you do not document the type of the excluded element.

Example:

```
     EXCLUDE SALARY.
```

B-F Take these paths to document the type of the element you are excluding. You specify the type for documentation purposes only.

Example:

```
     EXCLUDE DATA SET BOOKS.
```

G Identifies a data base element by a <DMS II name>. This name must reference a valid B 1000 Series DMS II element. Once you exclude a data base element, you cannot reference that element in the report specifications.

If <DMS II name> is a data set, it must be an embedded data set. All elements in an excluded data set, including any embedded data sets, are deleted from the vocabulary.

If <DMS II name> is a group item, all subordinate elements are deleted from the vocabulary.

If <DMS II name> is a set or subset, it is marked as excluded in the vocabulary file.

If <DMS II name> is an item, any group which contains the item is excluded from the vocabulary.

H Take this path if you do not require <qualification> to specify the <DMS II name>.

I Take this path if you require <qualification> to specify the <DMS II name>.

For a detailed description of <qualification>, see Section 2.

REDEFINE Statement

Use the REDEFINE statement to reenter a disjoint data set in the vocabulary under an alias name. With the REDEFINE statement, you can create another description of a data set under a different name. You can change or exclude any item in the data set by following the REDEFINE statement with the necessary ASSIGN or EXCLUDE statement.

You can also use multiple REDEFINE statements referencing the same data set to create several different descriptions of the data set under different names.

To use the REDEFINE statement, you must have previously added the data set to the vocabulary via a DATASET statement. No other DATASET statements referencing additional disjoint data sets can be made before the REDEFINE statement. The syntax for the REDEFINE statement is as follows:

```
                                          A
--->REDEFINE-----------------------> <data-set name> ---> (1)
               |              |
               +--->DATASET-->+


                         B
(1) --->AS-----------> <name> -----------------------> (2)


       C
(2) ------------------------------------------------------> .
      |                                                  |
      | D                     E          H               |
      +--->TOTAL-POPULATION---------------> <integer> ->+
                              |          |
                              | F        |
                              |--->IS--->|
                              |          |
                              | G        |
                              +---> = -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <data-set name> identifying the data set must be a valid B 1000 Series DMS II disjoint data set.
B	The <name> becomes the alias name of the duplicate data set.
C	Take this path if you use the default TOTAL POPULATION of 9999. The REPORTER III system uses TOTAL POPULATION to determine editing pictures for the reporting of statistics on items in the redefined data set.
D	Take this path to override the default TOTAL POPULATION.
E-G	These paths are equivalent.
H	The <integer> specifies the TOTAL POPULATION.

Example:

```
        .
        .
        .
   DATASET UNIV-PERSONNEL
     TOTAL-POP = 997.
   ASSIGN PIC "$ZZ,ZZ9.99" TO SALARY.
   REDEFINE UNIV-PERSONNEL AS UNIV-SUPR
     TOTAL-POP = 200.
   EXCLUDE SALARY.
        .
        .
        .
```

B 2000/B 3000/B 4000 SERIES OF SYSTEMS
DMS II LANGUAGE STATEMENTS

This section describes how to add one or more B 2000/B 3000/ B 4000 DMS II data bases to the vocabulary.

Example:

```
DB UNIV.
END DB.
```

This example adds all elements in the data base named UNIV to the vocabulary. The following is an overview of the syntax (for more detail refer to the individual statement descriptions):

```
   +<------------------------------------------------------- (1)
   |
   | A                      B
-------> <DB statement> ---------------------------------> (2)
                          |                               |
                          | C                             |
                          +---> <data-base information> ->+

(1) <------------------------+
                            E|
     D                       | F
(2) ---> <END DB statement> ------>
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <DB statement> identifies the DMS II dictionary files known as the DB-INVOKE files. These files are created by a utility program called RP3VDM. You must run RP3VDM once for each B 2000/B 3000/B 4000 DMS II logical or physical data base that RP3VOC uses. Run RP3VDM as a separate job prior to executing RP3VOC. For further information, refer to Section 7.

B Take this path to create a default vocabulary for the data base. RP3VOC automatically adds all elements of the data base to the vocabulary.

Default editing pictures are created for all items. The editing pictures are based on item attributes specified in DASDL.

A default TOTAL POPULATION of 9999 is assigned to all data sets. You use TOTAL POPULATION to determine default editing pictures. You use the editing pictures to print out statistical summaries in reports.

C Take this path when you want to use non-default information. You can specify how elements of the data base should be added to the vocabulary. This specification can change names, exclude certain elements, assign editing pictures, and change TOTAL POPULATION.

Example:

```
DB CUST-ACCT-INFO.
    DATA SET ACCTS-RECV TOTAL-POP = 999.
    ASSIGN EDITING PICTURE "99/99/99" TO
       DUE-DATE.
    EXCLUDE DATA SET ACCTS-CREDIT.
    END DB.
```

This example assigns a TOTAL POPULATION to ACCTS-RECV and an editing picture to DUE-DATE, but excludes the data set ACCTS-CREDIT and all elements in it. All other elements in the data base CUST-ACCT-INFO are added to the vocabulary by default.

D The <END DB statement> signals the end of specifications for the data base. All remaining elements in the data base are added by default to the vocabulary.

E Take this path to add additional data bases to the vocabulary.

Example:

```
DB UNIV.
END DB.
DB LOG-UNIV1 OF UNIV DB-INVOKE "LOGUDB".
END DB.
```

This example adds all elements in the physical data base UNIV and the logical data base LOG-UNIV1 to the vocabulary.

F Take this path when you have finished specifying all data bases.

RP3VOC SPECIFICATION EXAMPLE FOR B 2000/B 3000/B 4000 SERIES DATA BASE

The following example gives you a complete statement-by-statement description of a vocabulary specification for a DMS II data base. The data base UNIV is used in the following vocabulary specification. Figure 4-4 illustrates the DB-INVOKE file listing generated by RP3VDM. This listing represents a DB-INVOKE file of the data base.

RP3VOC accesses items in the data base in the sequential order displayed in the DB-INVOKE file listing. (See Figure 4-4.) Once RP3VOC processes an item, it cannot reference the item again in the vocabulary specification. The following are the vocabulary specifications:

```
DB UNIV.
    ASSIGN EDITING PICTURE "Z(3)9" TO ROOM.
    EXCLUDE INSTRUCTOR FROM VOCABULARY.
    DATASET BOOKS TOTAL-POP = 99999.
    DATASET UNIV-PERSONNEL TOTAL-POP= 999.
    SET UNIV-PERS-SSNUM AS UP-SSNUM.
    ASSIGN PIC "999I-99I-9999" TO SSNUM.
    REDEFINE UNIV-PERSONNEL AS PROFS
        TOTAL-POP 100.
    END DB.
```

The DB UNIV statement causes RP3VOC to initiate several actions. RP3VOC looks for the DB-INVOKE file UNIVOC on disk. UNIVOC is a list of all the elements in the data base corresponding to Figure 4-4. If this file is not on disk, the program will wait and require operator intervention. Running the RP3VDM program creates the DB-INVOKE file.

The ASSIGN statement causes RP3VOC to begin sequentially processing the elements of the data base until RP3VOC finds ROOM. All the elements preceding ROOM are added by default to the vocabulary with RP3VOC supplied default editing pictures. You cannot reference any of these items in subsequent vocabulary specifications.

Use the ASSIGN statement to add ROOM to the vocabulary with an editing picture of Z(3)9. ROOM is printed with this picture in reports.

The EXCLUDE statement causes RP3VOC to exclude the manual subset INSTRUCTOR, including all its items and sets, from the vocabulary. INSTRUCTOR is not added to the vocabulary, and cannot be referenced in report specifications. All data items following ROOM and preceding INSTRUCTOR are added by default to the vocabulary.

The first DATASET statement causes RP3VOC to add the embedded data set BOOKS to the vocabulary with an assigned TOTAL-POPULATION of 99999.

The next DATASET statement causes RP3VOC to add UNIV-PERSONNEL with an assigned TOTAL POPULATION of 999. A DATASET statement referencing the data item must precede the REDEFINE statement. The remaining elements of UNIV-COURSES, which are items of the embedded data set BOOKS, are added to the vocabulary by default.

The SET statement changes the name of UNIV-PERS-SSNUM to UP-SSNUM.

The ASSIGN statement causes RP3VOC to add SSNUM to the vocabulary with an editing picture of 999I-99I-9999. All elements of UNIV-PERSONNEL preceding SSNUM are added to the vocabulary by default.

The REDEFINE statement causes RP3VOC to create another description of the data set UNIV-PERSONNEL under the name PROFS. UNIV-PERSONNEL and its associated data elements are duplicated in the data set PROFS. PROFS is assigned a TOTAL POPULATION of 100.

RP3VOC can now reprocess the elements of the data set PROFS. In this example, however, no such references occur. When RP3VOC detects the END DB statement, all elements of the disjoint data set UNIV-PERSONNEL (alias PROFS) are added to the vocabulary by default.

Qualification by the name PROFS or UNIV-PERSONNEL is now required in report specifications to identify elements uniquely within these two data sets.

The END DB statement tells RP3VOC that this is the end of the data base specifications. All remaining elements of the DB-INVOKE data-base description are then added to the vocabulary by default.

```
 DB    UNIV ALL.
* 01     UNIV-COURSES STANDARD DATA SET (#3).
*        UNIV-COURSES SET SET (#7) OF UNIV-COURSES KEY IS
*             COURSE-ID.
*        UNIV-COURSES-LOC SET (#8) OF UNIV-COURSES KEYS ARE
*             BUILDING, ROOM.
*        02   COURSE-ID.
*             03    DEPARTMENT     PIC X(4)
*             03    LEVEL          PIC 999 COMP.
*        02   DAYS-OF-WEEK FIELD.
*             03    MON    BOOLEAN.
*             03    TUES   BOOLEAN.
*             03    WEDS   BOOLEAN.
*             03    THURS  BOOLEAN.
*             03    FRI    BOOLEAN.
*             03    SAT    BOOLEAN.
*        02   BUILDING          PIC XXX.
*        02   ROOM              PIC 9(4) COMP.
*        02   COURSENAME        PIC X(24).
*        02   CREDIT-HOURS      PIC 9(4) COMP.
*        02   CLASS-SIZE        PIC 99 COMP.
*        02   INSTRUCTOR MANUAL SUBSET (#4) OF UNIV-PERSONNEL
*             KEY IS NAME.
*        02   BOOKS STANDARD DATA SET (#5).
*             BOOK-SET SET (#6) OF BOOKS KEY IS BOOK-AUTHOR.
*             03   BOOK-TITLE     PIC X(50).
*             03   BOOK-AUTHOR    PIC X(15).
* 01     UNIV-PERSONNEL STANDARD DATA SET (#9).
*        UNIV-PERS-SSNUM SET (#11) OF UNIV-PERSONNEL KEY IS
*             SSNUM.
*        UNIV-PERS-NAME SET (#12) OF UNIV-PERSONNEL KEY IS
*             NAME.
*        UNIV-PERS-DEPT SET (#13) OF UNIV-PERSONNEL KEY IS
*             DEPT.
*        02   NAME.
*             03    LASTNAME       PIC X(20).
*             03    FIRSTNAME      PIC X(10).
*        02   SEX    PIC X.
*        02   SSNUM  PIC 9(9) COMP.
*        02   DEPT   PIC X(4).
*        02   COURSES MANUAL SUBSET (#10) OF UNIV-COURSES
*             KEY IS COURSE-ID.
```

Figure 4-4. RP3VOC Specifications Sample

The following is a list of the vocabulary of names and associated editing pictures produced by the RP3VOC specification:

UNIV	
UNIV-COURSES-SET	
UNIV-COURSES-LOC	
COURSE-ID	
DEPARTMENT	X(4)
LEVEL	Z(3)
DAYS-OF-WEEK	Z(2)
MON	X(5)
TUES	X(5)
WEDS	X(5)
THURS	X(5)
FRI	X(5)
SAT	X(5)
BUILDING	X(3)
ROOM	Z(3)9
COURSENAME	X(24)
CREDIT-HOURS	Z(4)
CLASS-SIZE	Z(2)
BOOKS	
BOOK-SET	
BOOK-TITLE	X(50)
BOOK-AUTHOR	X(15)
UNIV-PERSONNEL	
UP-SSNUM	
UNIV-PERS-NAME	
UNIV-PERS-DEPT	
NAME	
LASTNAME	X(20)
FIRSTNAME	X(10)
SEX	X
SSNUM	999I-99I-9999
DEPT	X(4)
COURSES	
UNIV-PERS-SSNUM	
UNIV-PERS-NAME	
UNIV-PERS-DEPT	
NAME	
LASTNAME	X(20)
FIRSTNAME	X(10)
SEX	X
SSNUM	Z(9)
DEPT	X(4)
COURSES	

DATA-BASE INFORMATION

The data-base information statements must access data-base items in the same order as these items occur in the DB-INVOKE listing. The DB-INVOKE listing is a COBOL listing generated by a DB INVOKE ALL. You obtain this listing from RP3VDM. The data-base information statements cause RP3VOC to process all elements of the DB-INVOKE file sequentially, until a specific element is referenced. These elements are added to the vocabulary as they are processed by RP3VOC.

RP3VOC enters all data-base elements sequentially. Unlike system files, you do not need to add B 2000/B 3000/B 4000 DMS II data items individually to the vocabulary. The entire data base or logical data base is added by default. Only specified data sets, sets, and data items are modified or excluded from the vocabulary. The following is a list of data-base information statements and their descriptions:

Statement	Explanation
ASSIGN Statement	Use the ASSIGN statement to specify an editing picture for a data item. The editing picture is used to print the data item in subsequent reports. If you do not specify an editing picture for a data item, RP3VOC automatically generates a default editing picture from the description of the item given in DASDL. Since the default editing picture may not provide the best form for printing, you can specify a new editing picture to override the default generated by RP3VOC.
DATASET Statement	The DATASET statement specifies a data set. You can use this statement three ways: 1. To assign a TOTAL POPULATION to a data set. The REPORTER III system uses TOTAL POPULATION to determine the number of spaces required for printing statistical summaries. Examples of statistical summaries are TOTAL and COUNT report specifications. 2. To give an alternate name to a disjoint data set.

3. To allow you to redescribe a disjoint data set using the REDEFINE statement. You must add a disjoint data set with the DATASET statement before you use the REDEFINE statement.

EXCLUDE Statement

Use the EXCLUDE statement to delete data sets and associated elements from the vocabulary. Use this statement to exclude sensitive or irrelevant items. Elements excluded from the vocabulary cannot be reported or referenced in report specifications.

REDEFINE Statement

Use the REDEFINE statement to reenter a disjoint data set into the vocabulary under a different name. The REDEFINE statement allows you to create a duplicate of the disjoint set. You can then redescribe elements of the disjoint set of the vocabulary while the original data-set descriptions remain under the original name. You must previously add the disjoint set to the vocabulary using the DATASET statement.

SET Statement

Use the SET statement to give an alternate name to a disjoint set.

ASSIGN Statement

Use the ASSIGN statement to specify an editing picture for a data item. This picture is used to print the data item in reports.

If you do not specify an editing picture for an elementary data item, RP3VOC generates a default editing picture from the description of the data item in the DB-INVOKE listing. The syntax for the ASSIGN statement is as follows:

```
----->ASSIGN------------------->PICTURE----------------> (1)
            |             |
            +-->EDITING-->+

                                    A
(1) --> "<COBOL picture>" -->TO---> <DMS II item name> --> (2)

       B
(2) ---------------------------> .
      |                        |
      | C                      |
      +---> <qualification> ->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <DMS II item name> must be an unsubscripted elementary DMS II data item.

For example, assume that a data item, DATE-NEEDED, with a NUMBER (6) description is used to store a date. The default editing picture is Z(6). If the printed image of DATE-NEEDED is to be MM/DD/YY, then the required result is obtained at report time by issuing the following statement to RP3VOC:

```
ASSIGN PIC "99I/99I/99" TO DATE-NEEDED.
```

Examples:

```
ASSIGN EDITING PICTURE "$Z(9).9(2)" TO
   SALARY.
ASSIGN PIC "999" TO AGE.
```

CAUTION

The COBOL editing picture cannot have any leading or trailing blanks. If the editing picture is smaller than the storage picture, truncation may result.

If you have set the option DECIMAL-POINT IS COMMA, then the appropriate picture must be given to avoid misalignment or COBOL syntax errors.

Examples:

If the option DECIMAL-POINT IS COMMA is not set,

```
ASSIGN PIC "ZZZ,ZZZ.99" TO QTY.
```

If the option DECIMAL-POINT IS COMMA is set,

```
ASSIGN PIC "ZZZ.ZZZ,99" TO QTY.
```

B Take this path if you do not require <qualification>.

C Take this path if you require <qualification> to specify the <DMS II item name>.

DATASET Statement

Use the DATASET statement to assign a TOTAL POPULATION to a data set or to give an alternate name to a disjoint data set. This statement also allows you to redescribe a disjoint data set using a REDEFINE statement. You must use a DATASET statement to add a disjoint data set to the vocabulary before you use a REDEFINE statement for it. The syntax for the DATASET statement is as follows:

```
             A
--->DATASET---> <data-set name> ----------------------> (1)


        B                          D
(1) ---------------------------------------------------> (2)
      |                       | |                    |
      | C                     | | E                  |
      +---> <qualification> ->+ +--->AS--> <name> ->+


        F
(2) -----------------------------------------------------> .
       |                                              |
       | G                  H       K                 |
       +---->TOTAL-POPULATION-------------> <integer> -->+
                            |        |
                            | I      |
                            |-->IS-->|
                            |        |
                            | J      |
                            +--> = ->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <data-set name> must be a valid B 2000/B 3000/B 4000 Series DMS II data set in the DB-INVOKE listing.
B	Take this path if you do not require <qualification> to specify the data set.

C Take this path if you require <qualification> to specify the data set.

D Take this path if you do not use an alternate name to identify the current description of the data set in the vocabulary. For an embedded data set, you must take this path.

Example:

```
DATASET UNIV-PERSONNEL.
```

E Take this path only if you are using an alternate name to identify a disjoint data set in the vocabulary description of the data set. The <name> you specify becomes the internal name for the data set in the generated report program. If you take this path, the <data-set name> in path A no longer exists in the vocabulary, and you cannot reference that name in report specifications.

Example:

```
DATASET UNIV-PERSONNEL AS PROFESSORS.
```

F Take this path if you use the default TOTAL POPULATION of 9999. The REPORTER III system uses TOTAL POPULATION to determine default editing pictures for printing statistical summaries.

G Take this path to override the default TOTAL POPULATION.

Examples:

```
DATASET BOOKS TOTAL-POP IS 1500.
DATASET UNIV-PERSONNEL AS PEOPLE
   TOTAL-POP 100.
```

H-J These paths are equivalent.

K The <integer> specifies the TOTAL POPULATION.

DB Statement

Use the DB statement to instruct RP3VOC that the specifications refer to a specific B 2000/B 3000/B 4000 DMS II data base. RP3VOC assumes that all statements in the vocabulary are descriptions of items in the specified data base until RP3VOC encounters an END DB statement.

See Section 7 for additional information on the DB-INVOKE file, and its creation by RP3VDM.

The syntax for the DB statement is as follows:

```
          A
--->DB----------------------------------------------> (1)
       |                                        |
       | B                                      |
       +---> <logical-data-base name> ->OF->+


                                         C
(1) ---> <physical-data-base name> ---------------------> (2)
                                      |              |
                                      | D            |
                                      +--->AS <name> ->+


        E
(2) ------------------------------------------------------> (3)
      |                                              |
      | F        G          I                        |
      +--->DDF--------------->PACK->"<pack ID>" ->+
                |          |
                | H        |
                +--->ON--->+


        J
(3) ------------------------------------------------------> (4)
      |
      | K               L
      +--->DB-INVOKE--------------------------------------> (5)
                     |                      |
                     | M                    |
                     +---> "<file name>" ->+
(4) ----------------------------------------------------------> .
                                                             |
         N                                                   |
(5) ------------------------------------------------------->|
     |                                                       |
     | O                        P                            |
     +--------------->PACK------------------------------>+
          |          |          |                        |
          +-->ON-->+           | Q                      |
                                +----> "<pack ID>" ->+
```

The paths of this syntax diagram are explained below.

Path	Explanation
A	Take this path to add a physical-data-base description to the vocabulary.
B	Take this path to add a logical-data-base description of a physical data base to the vocabulary. If you use this path, you must also use paths K and M. Example: DB LOG-UNIV1 OF UNIV DB-INV "LOGUDB".
C	Take this path if you use the actual data-base name rather an internal (or alias) name. The name of the data base in the vocabulary will be either the physical-data-base name specified in path A, or the logical-data-base name given in path B.
D	Take this path if you are specifying an internal (or alias) name for the data base. The name of the data base in the vocabulary will be the <name> you specify here. You can add a data base to the vocabulary a number of times by using internal names. Examples: DB UNIV AS ALIAS-UNIV. DB LOG-UNIV OF UNIV AS ALT-UNIV-NAME DB-INVOKE "LOGUDB".

E Take this path if the data-base-description (DDF) file is not on a restricted disk pack (that is, if the compiler needs no file equate to find the DDF file.)

F Take this path if the data-base-description (DDF) file is on a restricted disk path.

G-H These paths are equivalent.

I Take this path to specify the <pack ID>. The <pack ID> indicates the location of the disk pack containing the data-base-description (DDF) file.

Example:

```
DB UNIV AS ALIAS-UNIV.
   DDF PACK "REPORT"
   DB-INVOKE "UNIVOC".
```

J Take this path only if you specified path A, and are using the default DB-INVOKE file name. The default DB-INVOKE file name is xxxVOC, where xxx is the first three characters of the physical-data-base name. The default device is disk.

K Take this path to use non-default DB-INVOKE file attributes. If you specified path B, you must use this path and path M.

L Take this path only if you specified path A, and use the default DB-INVOKE file name of xxxVOC, where xxx is the first three characters of the physical-data-base name.

M Take this path to specify the <file name> containing the DB-INVOKE information. RP3VDM builds a DB-INVOKE file for each logical and physical data base you want added to the vocabulary. The DB-INVOKE file contains all of the elements in the logical or physical data base. You must take this path if you specified path B.

Example:

```
DB LOG-UNIV1 OF UNIV  DB-INVOKE "LOGUDB".
```

N Take this path for the default hardware type of disk.

O Take this path if the DB-INVOKE file is on disk pack. |

P Take this path to specify the default disk pack family name of PACK. |

Q Take this path to specify a disk pack family name other than the default. |

Example:

```
DB LOG-UNIV1 OF UNIV DB-INV "LOGUDB"
  PACK "REPORT".
```

END DB Statement

The END DB statement instructs RP3VOC to add any remaining elements of the DB-INVOKE file to the vocabulary. Each DB statement must have a corresponding END DB statement. The syntax for the END DB statement is as follows:

```
--------->END DB------------------> .
```

Example:

```
     END DB.
```

EXCLUDE Statement

Use the EXCLUDE statement to exclude data sets and associated elements from the vocabulary. By default, all elements in the data base are added to the vocabulary unless they are explicitly excluded. Use this statement to exclude sensitive and irrelevant information. The syntax for the EXCLUDE statement is as follows:

```
                A                I
-->EXCLUDE---------------------> <DMS II name> -> (1)
              |                  |
              | B                |
              |-->DATA SET------>|
              |                  |
              | C                |
              |-->SET----------->|
              |                  |
              | D                |
              |-->SUBSET-------->|
              |                  |
              | E                |
              |-->LINK ITEM----->|
              |                  |
              | F                |
              |-->CONTROL ITEM-->|
              |                  |
              | G                |
              |-->DATA ITEM----->|
              |                  |
              | H                |
              +-->GROUP ITEM--->+

           J
(1) ---------------------------------------------> (2)
         |                          |
         | K                        |
         +---> <qualification> ->+

(2) ---------------------------------------------> .
       |                      |
       +-->FROM VOCABULARY-->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if you do not document the type of the excluded element. Example: `EXCLUDE SSNUM`
B-H	Take these paths to document the type of the element you are excluding. You specify the type for documentation purposes only. Example: `EXCLUDE DATA SET UNIV-COURSES.`
I	Identify a data base element by a <DMS II-name>. This name must reference a valid B 2000/B 3000/B 4000 DMS II element described in the DB-INVOKE file. Once you exclude a data base element, you cannot reference that element in the report specifications. If a <DMS II name> is a data set, all elements in the data set, including any embedded data sets, are deleted from the vocabulary. If <DMS II name> is a group item, all subordinate elements are deleted from the vocabulary. If <DMS II name> is a set or subset, it is marked as excluded in the vocabulary file.
J	Take this path if you do not require <qualification> to specify the <DMS II name>.
K	Take this path if you require <qualification> to specify the <DMS II name>.

REDEFINE Statement

Use the REDEFINE statement to reenter a disjoint data set in the vocabulary under an alias name. With the REDEFINE statement, you can create another description of a data set under a different name. You can change or exclude any item in the data set by following the REDEFINE statement with the necessary ASSIGN, EXCLUDE, or SET statement.

You can also use multiple REDEFINE statements referencing the same data set to create several different descriptions of the data set under different names.

To use the REDEFINE statement, you must have previously added the data set to the vocabulary via a DATASET statement. No other DATASET statements referencing disjoint data sets can be made before the REDEFINE statement. The syntax for the REDEFINE statement is as follows:

```
                                   A
-->REDEFINE----------------------> <data-set name> ---> (1)
                |          |
                +-->DATASET-->+

                      B
(1) --->AS------------> <name> -----------------------> (2)

       C
(2) --------------------------------------------------->.
      |                                               |
      | D                     E         H             |
      +--->TOTAL-POPULATION---------------> <integer> -->+
                              |         |
                              | F       |
                              |-->IS--->|
                              |         |
                              | G       |
                              +--> = -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <data-set name> identifying the data set must be a valid B 2000/B 3000/B 4000 DMS II disjoint data set described in the DB-INVOKE file.
B	The <name> becomes the alias name of the duplicate data set.
C	Take this path if you use the default TOTAL POPULATION of 9999. The REPORTER III system uses the default TOTAL POPULATION to determine editing pictures for the reporting of statistics on items in the redefined data set.
D	Take this path to override the default TOTAL POPULATION.
E-G	These paths are equivalent.
H	The <integer> specifies the TOTAL POPULATION.

Example:

```
DATASET UNIV-COURSES.
ASSIGN PIC "Z9" TO CLASS-SIZE.
REDEFINE UNIV-COURSES AS UC-REDEF
  TOTAL-POP 10000.
EXCLUDE SET UNIV-COURSES-LOC.
```

SET Statement

Use the SET statement to change the name of a particular disjoint set. Only use a SET statement to resolve a possible conflict in names. The <set name> must be a valid DMS II disjoint set described in the DB-INVOKE file. The syntax for the SET statement is as follows:

```
                              A
-->SET--> <set name> -------------------------------> (1)
                         |                        |
                         | B                      |
                         +---> <qualification> ->+


     C
(1) --->AS-----> <name> ---------------------------> .
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if you do not require <qualification> to specify the disjoint set.
B	Take this path if you require <qualification> to specify the disjoint set.
C	The <name> becomes the internal name of the set. You can no longer use the <set name> to reference the disjoint data set in the report specifications.

Example:

```
SET UNIV-PERS-DEPT AS UP-DEPT.
```

SECTION 5

SYSTEM FILE LANGUAGE STATEMENTS

In this section, you will learn how to specify the location of system file descriptions, how to modify the descriptions, and how to add system file descriptions to a vocabulary. Section 5 first gives you an overview of system file descriptions, then provides detailed explanations and syntax diagrams for the system-file statements. At the end of the section are tables of COBOL source storage media and file attributes for the A Series, B 1000, and B 2000/B 3000/B 4000 Series of Systems.

Example:

```
SOURCE FILE "PAYS" IS ON DISK.
FILE PAYROLL.
  B IS EXEMPTIONS.
END FILE.
```

The SOURCE statement tells RP3VOC that the file and record descriptions may be found in the COBOL source file "PAYS". The FILE statement is used to add the system-file description, PAYROLL, to the vocabulary. The next statement changes the name of data item B to EXEMPTION in the vocabulary. The END FILE statement causes RP3VOC to add the remaining items in PAYROLL to the vocabulary by default.

The following is an overview of the syntax (for more detail refer to the individual statement descriptions):

```
 +<--------------------------------------------------------------+
 |                                                             G |
 |                                            +<----------------+ |
 |                                            |               E | |
 | A                  B                       | D               | | F
-----> <SOURCE   -----------------------------> <file       --------->
        statement> |                        |    specification>
                   | C                      |
                   +--> <COBOL source> -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The SOURCE statement identifies the COBOL source containing the description of one or more system files you want added to the vocabulary.
B	Take this path if the SOURCE statement in path A specifies the <external file name> and the storage medium of the COBOL source file. Use this path if you do not want to enter COBOL source in-line with the vocabulary specification.
C	Take this path if the SOURCE statement in path A indicates that the COBOL source follows. The <COBOL source> is the COBOL source code you want to enter in-line with the specification.
D	The system file and any modifications to its description are described in the <file specification>. The description of the file is extracted from the current COBOL source and added to the vocabulary with the desired changes.
E	Take this path if you want to add more system files from the current COBOL source. These system files must be specified in the same order their COBOL FDs occur in the COBOL source. RP3VOC treats the COBOL source as a sequential file; it cannot access file descriptions or data names that have already been processed.

Example:

```
SOURCE FILE "CSTSRC" IS ON DISK.

FILE REMIT-FILE.
END FILE.

FILE ACCT-FILE.
END FILE.
```

The FD and record descriptions of both "REMIT-FILE" and "ACCT-FILE" are extracted from the same COBOL source "CSTSRC". In "CSTSRC", the FD of "REMIT-FILE" must occur before that of "ACCT-FILE".

For more examples, see the SOURCE statement description.

F Take this path if no more system files are extracted from the current COBOL source.

G Take this path if more system files are extracted from another COBOL source and added to the vocabulary.

H Take this path when you have finished specifying all system files.

Example:

```
SOURCE FILE "XYZ" IS ON TAPE.
FILE STATE-VEHICLE-RECORDS.
% THE NEXT FOUR STATEMENTS MODIFY
%  THE RECORD DESCRIPTION
     LICENSE-NR IS LICENSE-NUMBER WITH
       PIC "XXX/XXX".
     FOUR-DOOR REFERS TO MODEL VA 4.
     TWO-DOOR REFERS TO MODEL VALUE 2.
     CONVERTIBLE REFERS TO MODEL VALUE 17.
END FILE.
SOURCE FILE "SOLD" IS ON DISK.
FILE STATE-EMPLOYEES IS EMPLOYEES.
     WAGE IS HOURLY-WAGE.
END FILE.
FILE EARNINGS.     % INCLUDE EARNINGS FILE
END FILE.
```

In this example, the source tape file "XYZ" contains the description of the file STATE-VEHICLE-RECORDS, and the source disk file "SOLD" supplies descriptions of the files STATE-EMPLOYEES and EARNINGS.

SYSTEM FILE SPECIFICATION EXAMPLE

The following example offers you a complete statement-by-statement description of how RP3VOC processes COBOL source for system files. The following is the vocabulary specification:

```
SOURCE FILE "PAYS" IS ON DISK.
FILE PAYROLL.
B IS EXEMPTIONS.
FILLER IS SALARY WITH PICTURE "$(5).99".
FILLER OF PERSONAL-DATA IS PERSON-AGE.
H IS FILLER.
END FILE.
FILE DEPT IS DEPARTMENT.
    .
    .
    .
```

The source file, PAYS, contains the following COBOL code (written in Medium Systems ANSI-74 COBOL):

```
IDENTIFICATION DIVISION.
     .
     .
     .
    FD CARD-FILE
       RECORDING MODE ...
     .
     .
     .
       DATA RECORD IS CARD-IMAGE.
    01 CARD-IMAGE.
       02 B                 PIC 99.
         .
         .
       02 Z                 PIC 9(5).

(1) FD PAYROLL
       VALUE OF FILENAME IS "PAYR"
       DATA RECORD IS PAY-REC.
    01 PAY-REC.
       02 A                 PIC 9(3).
(2)    02 B                 PIC 99.
       02 C
         03 D               PIC 99V99.
(3)      03 FILLER          PIC 9(5)V99.
         03 FILLER          PIC XX.
       02 E                 PIC A(6).
```

```
           02 F                PIC X(20).
           02 PERSONAL-DATA.
             03 G              PIC 9(4) COMP.
     (4)     03 FILLER         PIC 99 COMP.
             03 FILLER         PIC 9.
     (5)     03 H              PIC X.
           02 I                PIC 9(3).
           02 J                PIC A(3).
        FD PRINT-FILE
              .
              .
              .
     (6) FD DEPT
              .
```

NOTE

"VALUE OF FILENAME IS" would be "VALUE OF TITLE IS" for B 1000 systems. On A Series systems it could be either.

Explanations of each statement in the preceding specification are given here:

SOURCE FILE "PAYS" IS ON DISK.

The SOURCE statement identifies the file "PAYS" as the COBOL source program to be examined by RP3VOC.

FILE PAYROLL.

The FILE statement instructs RP3VOC to find the SELECT statement and FD named PAYROLL, indicated by (1), and to store the file description in the vocabulary files. Since PAYROLL follows CARD-FILE in the source, you cannot reference CARD-FILE. If CARD-FILE were to be included, a FILE CARD-FILE and END FILE statement would have to precede the FILE PAYROLL statement.

B IS EXEMPTIONS.

The next statement, B IS EXEMPTIONS, causes RP3VOC to change the name of item B, indicated by (2), to EXEMPTIONS. The following is added to the vocabulary:

```
02 EXEMPTIONS       PIC 99.
```

PAY-REC and A are added to the vocabulary by default. Since RP3VOC is positioned on (2), you can no longer reference PAY-REC and A in the specification.

FILLER IS SALARY WITH PICTURE "$(5).99".

This statement causes RP3VOC to add C and D to the vocabulary by default. RP3VOC recognizes the COBOL reserved word FILLER. The first FILLER, indicated by (3), is changed to:

```
03 SALARY           PIC 9(5)V99.
```

SALARY, along with the editing picture $(5).99, becomes an element of the vocabulary. Only the name changes on the COBOL statement. SALARY is stored in the data file with the original editing picture. The editing picture is used for the output of SALARY in reports. Note that the second FILLER is not affected by the RP3VOC statement.

FILLER OF PERSONAL-DATA IS AGE.

The statement FILLER OF PERSONAL-DATA causes RP3VOC to add E, F, PERSONAL-DATA, and G to the vocabulary by default. Qualification is used in this statement two ways:

1. To uniquely identify a name.
2. To position RP3VOC at the proper element.

The 03 FILLER, indicated by (4), is changed to the following:

```
03 PERSON-AGE           PIC 99 COMP.
```

Thus, PERSON-AGE becomes an element of the vocabulary.

If you do not rename FILLER fields in the specification, the FILLER is added to the vocabulary, but the REPORTER III system cannot access these FILLER fields for reporting. For example, the FILLER following (4) is not accessible for reporting.

If, for example, you wanted to rename the FILLER following (4) without changing th_ FILLER at (4), you could use any of the following statements:

```
FILLER OF PERSONAL-DATA.
FILLER IS ABC.  % THIS SPECIFIES A NAME FOR THE
                % 2ND FILLER.
```

(or)

```
FILLER OF G.
FILLER IS ABC.

          (or)

FILLER OF FILLER OF G IS ABC.
```

The preceding examples illustrate that RP3VOC qualification can be used to both position the RP3VOC system ahead in the source file and to resolve ambiguities between data items.

H IS FILLER.

This statement protects H from being accessed by the REPORTER III System. Report Language statements cannot access the data in H because the word FILLER is added to the vocabulary instead of H. The COBOL source statement, indicated by (5), is changed to:

```
03 FILLER           PIC X.
```

Renaming an item FILLER is a simple way of protecting sensitive information from the REPORTER III System.

END FILE.

The END FILE causes RP3VOC to add the remaining data names of the file description (I and J) by default to the vocabulary. RP3VOC will continue adding data names to the vocabulary until one of the following COBOL constructs is encountered:

1. An FD.
2. A DATA-BASE SECTION.
3. A WORKING-STORAGE SECTION
4. A PROCEDURE DIVISION.

FILE DEPT IS DEPARTMENT.

The final statement FILE DEPT IS DEPARTMENT causes RP3VOC to search for an FD named DEPT. Upon finding the FD DEPT at (6), RP3VOC changes the file name DEPT to DEPARTMENT in the vocabulary. Note that no statements in the specification refer to any data names in the FD PRINT-FILE; therefore, no data names in PRINT-FILE are added to the vocabulary.

The following is a list of the vocabulary of names and the related output editing pictures produced by the vocabulary specification:

```
PAYROLL
PAY-REC
A                                                 Z(3)
EXEMPTIONS                                        Z9
C
D                                                 Z9.99
SALARY                                            $(5).99
E                                                 A(6)
F                                                 X(20)
PERSONAL-DATA
G                                                 Z(4)
PERSON-AGE                                        Z9
I                                                 Z(3)
J                                                 A(3)
DEPARTMENT
   .
   .
   .
```

The COBOL codes placed in the vocabulary file and used as input to the REPORTER III System are as follows:

```
FD PAYROLL
  VALUE OF TITLE IS "PAYR"     [VALUE OF FILENAME IS "PAYR"]
  DATA RECORD IS PAY-REC.      [A Series, B 2000-B4000]
  01 PAY-REC.
  02 A                  PIC 9(3).
  02 EXEMPTIONS         PIC 99.
  02 C.
     03 D               PIC 99V99.
     03 SALARY          PIC 9(5)V99.
     03 FILLER          PIC XX.
  02 E                  PIC A(6).
  02 F                  PIC X(20).
  02 PERSONAL-DATA.
     03 G               PIC 9(4) COMP.
     03 PERSON-AGE      PIC 99 COMP.
     03 FILLER          PIC 9.
     03 FILLER          PIC X.
  02 I                  PIC 9(3).
  02 J                  PIC A(3).
FD DEPARTMENT
     .
     .
     .
```

FILE SPECIFICATION

File specification is the combination of statements used to point RP3VOC to specific data-structure descriptions located in a COBOL source program. These statements allow you to extract the COBOL source code and make changes to the description. You must have previously specified the source program using a SOURCE statement.

The following is the general syntax for file specification (for more detail refer to the individual statement descriptions):

```
              A
--------------> <FILE statement> ----------------> (1)


        B                                       G
(1) ------------------------------------------------> (2)
        |                                       |
        |                                     F |
        |<--------------------------------------|
      C |                                       |
        | D                                     |
        |---> <Data-name-change statement> --->|
        |                                       |
        | E                                     |
        +---> <Condition-addition statement> ->+


      H
(2) ---> <END FILE statement> ---------------->
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <FILE statement> identifies the file description you want added to the vocabulary.
B	Take this path if you are not modifying the record descriptions contained in the COBOL source. All elements in the description are added by default to the vocabulary. Names are entered into the vocabulary as they appear in the COBOL

source description. Default editing pictures are assigned based on the storage pictures.

C Take this path to modify elements of record description in the COBOL source. You must specify elements in the same order as they appear in the COBOL source. The item name may be changed or removed, and an editing picture and/or conditions assigned to an item. If you use an editing picture, it must be enclosed in quotes with no leading or trailing spaces. If the editing picture is smaller than the storage picture, results within the REPORTER III generated reports might be truncated.

D Use the <Data-name-change statement> to assign a new name and/or editing picture to the COBOL description. The editing picture will be used for reporting the item. You may also use this statement to effectively delete an item from the description.

Example:

<u>X3</u> <u>IS</u> <u>HOURS-WORKED</u> <u>WITH</u> <u>PIC</u> <u>"ZZ9.99".</u>

The name of COBOL elementary item X3 is changed to HOURS-WORKED and is assigned the editing picture "ZZ9.99".

E Take this path to add a condition for a COBOL elementary item in the record description.

Example:

<u>RED</u> <u>REFERS</u> <u>TO</u> <u>COLOR</u> <u>WITH</u> <u>VALUE</u> <u>3.</u>

The 88-level condition RED is associated with the COBOL elementary item COLOR.

F Use this path to specify additional modifications to elements in the record description. You must specify these modifications in the same order as the items occur in the COBOL source, since RP3VOC references items sequentially. Once it positions itself at a COBOL item, RP3VOC cannot reference preceding COBOL source items.

Example:

Items in COBOL Source	Sample Vocabulary Statements
.	.
.	.

```
      .                  .
  02 SAGE PIC 99.      SAGE IS STUDENT-AGE

  02 SEX PIC 9.        SEX IS FILLER.

  02 SAL 9(5)V99.      SAL IS SALARY WITH PIC "$(6).99".
      .                  .
      .                  .
      .                  .
```

The corresponding modification statements are supplied in the same order as the items SAGE, SEX, and SAL appear in the COBOL source. Once it positions itself at SAL, for example, RP3VOC cannot reference preceding COBOL source items such as SAGE or SEX.

G Take this path if you are finished modifying the record descriptions.

H The <END FILE statement> signals the end of specification for a specific data structure. All elements in the COBOL record descriptions for the file are added by default to the vocabulary unless these elements are specifically modified.

CONDITION-ADDITION STATEMENT

The Condition-addition statement allows you to describe a condition for the COBOL <data name> of an existing COBOL data item. RP3VOC automatically generates a COBOL condition name for the specified <data name>.

You can add conditions to COBOL <data name>s that already have COBOL condition names and to COBOL elementary data items that do not currently have conditional variables.

The vocabulary statement builds the COBOL condition name and enters the <condition name> in the vocabulary. Several new conditions can be generated for the same COBOL <data name>. The syntax for the Condition-addition statement is as follows:

```
                                              A
----> <condition name> -->REFERS--------------> <data name> -> (1)
                                  |          |
                                  +-->TO-->+


      B
(1) ------------------------------------------->VALUES-----> (2)
      |                          |   |          |
      | C                        |   +-->WITH-->+
      +--> <qualification> -->+


      +---------------------- , <-------------------+
      |                                            J|
      | D                  E                        | K
(2) ------> <literal> ------------------------------------> .
                     F|                             |
                      | G            I              |
                      |-->THRU---------> <literal>-->+
                      |              |
                      | H            |
                      +-->THROUGH-->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	The <data name> instructs RP3VOC where to insert the condition in the COBOL source code. If you want to change a <data name>, you must use the DATA-NAME-CHANGE statement before you add a condition. Use the new data name as the <data name> in this statement.

For example, RP3VOC is processing the following COBOL source statements:

```
        .
        .
    02 TYPE          PIC 9.
    02 CODE.
       03 FIRST      PIC 9(3).
            .
            .
```

Assume you want to change the <data name> TYPE to COMPANY-TYPE, and add two conditions. The following three RP3VOC statements give the required results:

```
     TYPE IS COMPANY-TYPE.
     INDUSTRIAL REFERS TO COMPANY-TYPE
       WITH VALUE 3.
     HOSPITAL REFERS TO COMPANY-TYPE VA 5.
```

Note that COMPANY-TYPE is used in the statements. The COBOL source code stored in the vocabulary files is as follows:

```
       .
       .
       .
    02 COMPANY-TYPE      PIC 9.
       88 INDUSTRIAL     VALUE 3.
       88 HOSPITAL       VALUE 5.
       .
       .
       .
```

B Take this path if no positional <qualification> is necessary.

C Take this path if <qualification> is required to position RP3VOC at the <data name>.

Example:

```
     GREEN REFERS TO COLOR OF EYES WITH VA 5.
```

D Condition values are specified using literals. A condition VALUE string literal cannot exceed 30 characters.

Example:

```
     DRACULA REFERS TO BLOODTYPE WITH
       VALUE "ABNEGATIVE".
```

E Take this path if you are not using the THROUGH option for the condition value(s). The THROUGH option defines the condition as a range of values.

F Take this path if you are using the THROUGH option to specify a range of condition values.

Example:

```
     FIRST-QTR REFERS TO MONTH WITH
       VALUES 01 THRU 03.
```

In this example, FIRST-QTR refers to MONTH with values of 1, 2, or 3.

G-H These paths are equivalent.

I The <literal> specified here must be the same type (number or string) as the <literal> specified in path D. This <literal> specifies the inclusive end of a range of condition values.

J Take this path to specify additional condition values or condition-value ranges.

K Take this path when you have finished specifying condition values.

Examples:

```
     RED REFERS TO COLOR WITH VALUE 3.
     PINK REFERS TO COLOR WITH VA 8
       THRU 11, 15, 17 THRU 20.
```

DATA-NAME-CHANGE STATEMENT

The Data-name-change statement allows you to replace an existing COBOL <data name> with a new name. This statement can also add an optional editing picture for the item. The syntax for the Data-name-change statement is as follows:

```
                           A
----> <data name> --------------------------------> (1)
                      |                        |
                      | B                      |
                      +--> <qualification> -->+


         C                         H
(1) -----------------------------------------------> (2)
       D|                        | |
        | E        G             | | I
        |--->IS-----> <name> -->+ +---->WITH--------> (3)
        |        |
        | F      |
        +--> = ->+


(2) --------------------------------------------------> .
                                                |
(3) ---------------------> "<COBOL picture>" -->+
      |             |
      |-->PICTURE-->|
      |             |
      +-->PIC------>+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if no <qualification> is required to specify the required <data name>. Example: SALARY.

This statement causes RP3VOC to continue scanning the COBOL source until it finds the <data name> SALARY. This statement only positions RP3VOC at the name of the COBOL source file.

B Take this path to qualify the <data name>. Qualification instructs RP3VOC to find higher level qualifiers in the COBOL source before finding the qualified <data name>. Qualifiers must therefore be presented in the sequence they appear in the file. You cannot reference an item in a qualification that has already been processed.

Example:

```
W OF COLOR OF EMPLOYEE IS WAGE.
```

This statement instructs RP3VOC to scan the COBOL source for EMPLOYEE, and next for COLOR, and finally for W.

C Take this path if you are not changing the data name. The name which appears in the COBOL record description is added to the vocabulary.

D Take this path to change the <data name>. Changing the <data name> has several purposes:

1. To give a more appropriate and meaningful name to an item. The <data name> is used as the default column heading in REPORTER III automatically formatted reports.
2. To protect sensitive data from being accessed and reported.
3. To rename data stored as FILLER, making it reportable.
4. To eliminate undesirable but required qualifications from report specifications.

The following three examples illustrate the different uses of the DATA-NAME-CHANGE statement:

Example:

```
X1 IS EMPLOYEE-NO.
```

This statement changes the <data name> to a more meaningful name. You can use the new name as a heading in the report.

Example:

```
     SEX IS FILLER.
```

This statement prevents SEX from being reported. Explicitly changing a <data name> to FILLER causes all groups and records which contain the <data name> to be non-reportable.

Example:

```
     FILLER IS INFO.
```

This statement allows the information in FILLER to be accessible and reportable as INFO.

E-F These paths are equivalent.

G The name specified here becomes the new name of the item.

H Take this path if no editing picture is specified. In this case, RP3VOC generates a default editing picture from the internal storage picture extracted from the record description.

I Take this path to specify an editing picture. The default editing picture may not provide the best form for printing in reports. The picture you specify here overrides the default editing picture generated by RP3VOC.

Example:

```
     BIRTH-DATE WITH PICTURE "99/99/99".
```

The <COBOL picture> must be given as a <string> with no leading or trailing spaces. If the editing picture is smaller than the storage picture, the results in the REPORTER III generated reports might be truncated. RP3VOC does not check the validity of the supplied <COBOL picture>, so be sure you specify a valid COBOL editing picture.

Examples:

If the option DECIMAL-POINT IS COMMA is not set,

```
W IS WAGE WITH PIC "$(3).99".
```

If the option DECIMAL-POINT IS COMMA is set,

```
W IS WAGE WITH PIC "$(3),99".
```

END FILE STATEMENT

The END FILE statement signals the end of a file specification. RP3VOC adds the remaining elements of the system file description to the vocabulary by default when it encounters the END FILE statement. The syntax for the END FILE statement is as follows:

```
---->END------------------> .
          |             |
          +-->FILE-->+
```

Example:

```
     END FILE.
```

Consider the COBOL source program as containing the following system file descriptions:

```
       .
       .
       .
FD PAYS
       .
       .
       .
   VALUE OF TITLE IS "PAYFIL".  [VALUE OF FILENAME IS "PAYFIL"]
01 PAY-RECORD.                  [A Series, B 2000-B 4000]
    02 EMPNO                PIC 9(6).
    02 NAME.
       03 LAST              PIC X(12).
       03 FIRST             PIC X(8).
       03 MI                PIC X.
    02 WAGE                 PIC 9(3)V99.
    02 EXP                  PIC 99.
FD CARD-FILE
          .
          .
          .
```

The vocabulary statements are as follows:

```
SOURCE FILE "PAYSOU" IS ON DISK.
FILE PAYS.
EMPNO IS EMPLOYEE-NO.
END FILE.
```

The resulting vocabulary consists of the following names:

PAYS	(FILE NAME)
PAY-RECORD	(RECORD NAME)
EMPLOYEE-NO	(DATA ITEM)
NAME	(GROUP NAME)
LAST	(DATA ITEM)
FIRST	(DATA ITEM)
MI	(DATA ITEM)
WAGE	(DATA ITEM)
EXP	(DATA ITEM)

The following example shows the file description "PAYS" being added to the vocabulary with no name changes.

```
SOURCE FILE "PAYSOU" IS ON DISK.
FILE PAYS.
END FILE.
```

FILE STATEMENT

The FILE statement identifies the file you want added to the vocabulary. You can use this statement to change the internal and/or external names of a system file.

Using this statement, you can also specify a TOTAL-POPULATION for the file. The TOTAL-POPULATION is used by REPORTER III to determine the number of spaces required for printing statistical summaries in reports, such as TOTAL and COUNT.

The FILE statement points RP3VOC to the description of the file specified by <file name>. The description for system files is the COBOL FD entry. RP3VOC processes the entire FD until it encounters one of the following COBOL constructs:

1. FD.
2. SD.
3. DATA-BASE SECTION.
4. WORKING-STORAGE SECTION.
5. PROCEDURE DIVISION.

The syntax for the FILE statement is as follows:

```
                                   A
---->FILE--> <file name> -------------------------------------------> (1)
                           |                                    |
                           | B   C         E                    |
                           +------->IS------> <file name> --->+
                              |         |
                              | D       |
                              +--> = -->+


          F
(1) --------------------------------------------------------------> (2)
     |
     | G               H                           K
     +----------------->IDENTIFICATION------------------> (3)
        |      |   |                      |    |           |
        +->WITH-->+ | I                    |    | L         |
                   |---->FILENAME-------->|    |-->IS---->|
                   |                      |    |           |
                   | J                    |    | M         |
                   +---->TITLE----------->+    +--> = -->+
                                                  O
(2) -----------------------------------------------------------> (4)
                                           |    |
    N                                      |    | P
(3) ---> <external file name> ->+    +--->TOTAL-POPULATION------> (5)


(4) ----------------------------------------> .
                                            |
       Q          T                         |
(5) ----------------> <integer>-->+
       |          |
       | R        |
       |--->IS-->|
       |          |
       | S        |
       +---> = ->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if you are not changing the <file name>. Example: FILE ORDER.
B	Take this path to change the name of a system file. Example: FILE PAY-FILE IS PAYROLL.
C-D	These paths are synonymous.
E	This <file name> specifies the new name used for the system file. Use this name to reference the file in the Report Language INPUT statement.

F Take this path if you are not specifying an <external file name>. The external file name is derived from the FD description for a system file.

For a system file with a "VALUE OF TITLE IS <data name>" (A Series and B 1000 Series), or "VALUE OF FILENAME IS <data name>" (A Series and B 2000-B 4000 Series), if you do not specify an external file name here, you must enter the <data name> and its value by a user-created COBOL input procedure.

If an external file name and pack name are specified in the FD, you can use this path to change the file name and retain the pack name by repeating the pack name in the clause.

Example:

```
FILE PAY WITH ID "DEF" PACK "XYZ".
```

The FD has the internal file name PAY and the external file name "ABC" on pack "XYZ". The file name is to be changed to "DEF" but the pack name remains "XYZ".

G Take this path to specify an <external file name>.

H-J These paths are equivalent. Use these options to a specify an <external file name> for a system file. The external file name specified here overrides the VALUE OF TITLE (for A Series and B 1000 Series) and VALUE OF FILENAME or VALUE OF FAMILYNAME (for A Series and B 2000-B 4000 Series) clauses in the COBOL FD.

Examples:

```
FILE PAY-FILE IS PAYROLL WITH ID "PAYS".
FILE PAY-FILE IS PAYROLL FILENAME "PAYTST".
```

K-M These paths are equivalent.

N Take this path if the <external file name> specifies both the <external file name> and the disk pack name for the system file.

Examples:

```
FILE PAY-FILE IS PAYROLL WITH ID "PAYS"
  WITH PACK "PAYRPK".
```

```
          FILE PAY-FILE IS PAYROLL FILENAME "PAYS"
            FAMILYNAME "PAYRPK".
```

O   Take this path if you are using the default TOTAL-POPULATION of 9999 records for the file.

P   Take this path to override the default TOTAL-POPULATION. The TOTAL-POPULATION is associated with all items in the file and determines the field size required by REPORTER III for printing statistical summaries on items in the file.

Example:

```
          FILE PAY-FILE TOTAL-POP = 50000.
```

For this example, reports referencing data items in this file will be allocated sufficient space to print summary statistics (count, total, and so forth) based on a total number of records of 50,000.

Q-S   These paths are equivalent.

T   The integer specifies the TOTAL-POPULATION of the file.

SOURCE STATEMENT

The SOURCE statement identifies the COBOL source which contains the file and/or record descriptions you want added to the vocabulary. The COBOL source must be a COBOL program having the following form:

```
[any legal COBOL construct preceding
FILE-CONTROL]

   FILE-CONTROL.
       [SELECT statements]

   DATA DIVISION.
       [FD(s) for system files]
   [DATA-BASE SECTION]

   WORKING-STORAGE SECTION.


[any legal COBOL construct following WORKING-
 STORAGE SECTION]
     PROCEDURE DIVISION.
```

System files require the COBOL statements FILE-CONTROL, DATA DIVISION, and PROCEDURE DIVISION. If you are using a system other than an A Series System, the COBOL source code must adhere to the position (margin) and specification rules of ANSI-74 COBOL. If you are using an A Series System, the COBOL source code must adhere to the rules of the version of COBOL that you are using, either ANSI-74 COBOL or ANSI-85 COBOL.

RP3VOC will continue to use the specified source file to look for data descriptions for each <FILE statement>, and END FILE statement. Any one of the following three vocabulary statements will cause RP3VOC to discontinue use of the current source file:

1. A SOURCE statement.
2. An INPUT statement.
3. An End-of-File (EOF).

RP3VOC first scans the COBOL source to gather any SELECT statements. RP3VOC then readies itself to process data descriptions specified in the vocabulary statements.

RP3VOC imposes a few general COBOL source language limitations. RP3VOC does not process the DATA DIVISION clause RENAMES; therefore, COBOL source statements containing the RENAMES clause are not added to the vocabulary. If you are using a system other than an A Series System, the COBOL source must conform to ANSI-74 COBOL standards. If you are using an A Series System, the source must conform to the standards of the version of COBOL that you are using, either ANSI-74 COBOL standards or ANSI-85 COBOL standards.

RP3VOC cannot analyze COBOL source statements which are continued over a line using a hyphen (-) in column 7. If a source statement contains a hyphen in column 7, an error occurs and RP3VOC deletes the statement. You must then reformat the source statement, eliminate continuation lines and rerun RP3VOC. All unrecognized COBOL constructs are documented in Appendices B, C, and D.

The COBOL verb COPY allows you to add library routines contained on source language library files to the COBOL source program. Restrictions on the use of the COPY verb with RP3VOC are:

1. The COPY statement must begin on a separate line.
2. If the REPLACING clause exists, it is ignored.

For more information on COPY restrictions, see Appendix B, C, or D. RP3VOC treats the COBOL text in library files the same as other COBOL statements, allowing you to change data names, add conditions, etc.

The syntax for the SOURCE statement is as follows:

```
                                 A
---->SOURCE---------------------->FOLLOWS---------------> (1)
          |        |   |
          +-->FILE-->+   | B
                       +-----> <external file name> ----> (2)


(1) ------------------------------------------------> .
                                                     |
                              C                      |
(2) ------------------------>CARDS------------------>+
     |     |   |     |   |                       |
     +-->IS-->+ +-->ON->+ | D                     |
                         |-->DISK--------------->|
                         |                       |
                         | E                     |
                         |-->LIBRARY------------>|
                         |                       |
                         | F                     |
                         |-->TAPE--------------->|
                         |                       |
                         | G                     |
                         +-->DISKPACK----------->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Use this path to add COBOL source in-line with RP3VOC statements. The COBOL source statements must immediately follow the SOURCE FILE FOLLOWS statement. This option allows you to access COBOL data descriptions not available on an existing COBOL source program or library file. You can also add necessary elements of a file description which do not reside in the COBOL source program, e.g., the specification of COBOL SELECT statement(s) for FD(s) existing library files.

Example:

```
SOURCE FILE FOLLOWS.
FILE-CONTROL.
   SELECT CARD-FILE ASSIGN TO READER.
   SELECT DISK-FILE ASSIGN TO DISK,
     ACCESS MODE IS SEQUENTIAL.
DATA DIVISION.
FD CARD-FILE
   VALUE OF TITLE       [VALUE OF FILENAME IS "TRANS"]
         IS "TRANS"     [A Series, B 2000 - B 4000]
   DATA RECORD IS CARD-IMAGE.
01 CARD-IMAGE.
   02 TYPE            PIC 99.
   02 FILLER          PIC X(8).
   02 TRANS-CODE      PIC 9(8).
   02 FILLER          PIC X(2).
   02 TRANS-TYPE      PIC 9(4).
   02 FILLER          PIC X(56).
FD DISK-FILE
   VALUE OF TITLE       [VALUE OF FILENAME IS "TRDSK"]
         IS "TRDSK"     [A Series, B 2000 - B 4000]
   DATA RECORD IS T-REC.
01 T-REC.
   02 NO              PIC 9(8) COMP.
   02 TDATE           PIC 9(6) COMP.
   02 FILLER          PIC X(5).
   02 DESCRIPTION     PIC A(20).
END-OF-JOB.
% VOCABULARY STATEMENTS PERTAINING TO SYSTEM FILE
% DESCRIPTIONS ARE GIVEN HERE
FILE CARD-FILE.
END FILE.
FILE DISK-FILE.
TDATE WITH PICTURE "99/99/99".
END FILE.
```

In this example the <COBOL source> consists of COBOL statements beginning with "FILE-CONTROL." and ending with "END-OF-JOB.". The following example shows how you use COBOL SELECT statements to access COBOL FDs residing in library files:

Example:

```
SOURCE FILE FOLLOWS.
FILE-CONTROL.
   SELECT PERSONNEL ASSIGN TO DISK.
   SELECT PAYROLL ASSIGN TO DISK
        ACCESS MODE IS RANDOM
        ACTUAL KEY IS PAYKEY.
DATA DIVISION.
FD PERSONNEL
COPY "LIBPER".
FD PAYROLL
COPY "LIBPAY".
END-OF-JOB.
FILE PERSONNEL.
END FILE.
FILE PAYROLL.
END FILE.
```

B Take this path if you do not want to add COBOL source in-line with vocabulary statements, i.e., if you are adding the COBOL source file to the vocabulary by identifying the <external file name> and storage medium. The <external file name> identifies the COBOL source file.

C-G Take these paths to specify the COBOL source storage media and file attributes for the different systems. Refer to Table 5-1 for A Series Systems, Table 5-2 for B 1000 Series Systems, or Table 5-3 for B 2000/B 3000/B 4000 Series Systems for more information.

Examples:

```
SOURCE FILE "CBSOLD" IS ON DISK.
SOURCE "CBSOLT" ON TAPE.
SOURCE "ILL005" IS LIBRARY.
SOURCE "FORTDB" IS ON DISK.
SOURCE "64FILE" IS ON CARDS.
```

Table 5-1

Storage Media and File Attributes
A Series Systems

Path	Storage Medium	File Attributes
C	Cards	80-column card file.
D	Disk	Card image disk file.
E	Library	COBOL library file.
F	Tape	Tape File.
G	Disk pack	Card-image file on disk pack.

Table 5-2

COBOL Source
Storage Media and File Attributes
B 1000 Series Systems

Path	Storage Medium	File Attributes
C	Cards	80- or 96-column card file.
D	Disk	Characters per record and blocking is assigned DEFAULT.
E	Library	Characters per record and blocking is assigned DEFAULT.
F	Tape	80-character/record tape file, blocked 9.
G	Disk pack	Characters per record and blocking is assigned DEFAULT.

Table 5-3

COBOL Source
Storage Media and File Attributes
B 2000/B 3000/B 4000
Series Systems

Path	Storage Medium	File Attributes
C	Cards	80-character card file.
D	Disk	80-character/record disk file, blocked 9.*
E	Library	COBOL 80-character/record disk file, blocked 9.**
F	Tape/SOLT	80-character/record tape file, blocked 9.
G	Disk pack	80-character/record disk-pack file, blocked 9.*

* For information on reblocking files, see the introduction in Appendix C.
** For more information, refer to the COPY statement in Appendix C.

SECTION 6

INPUT PROCEDURES

Section 6 first outlines the requirements and procedures for creating input procedures not provided by REPORTER III System input routines. The section then gives a detailed example of the use of an input procedure, and provides information on the COBOL source code and file input processes. The latter part of the section includes syntax diagrams and expanded explanations of input statements and specifications.

The report portion of the REPORTER III System generates input routines capable of obtaining most types of input. Sometimes these routines cannot access certain input. For example, you cannot use generated input routines for any of the following cases:

1. To distinguish between two similiar files.

2. To add default values to data items.

3. To add conditions that determine if data is used as input for the report.

In general, you would create an input procedure any time a report-generated input routine does not satisfy your needs.

Before a report can be generated, the input procedure must produce a logical record which describes all the information required to generate the report. Generated input routines associate each logical record with one physical record, and the program accesses and inputs one record at a time. If you want several physical records to be accessed and input as one logical record, you must create an input procedure.

You can use an unlimited number of input procedures. You can supply several different input procedures for a data structure or group of structures. In addition, if you are using a system other than an A Series System, any COBOL source used in the input procedure must conform to ANSI-74 COBOL standards. If you are using an A Series System, the COBOL source used in the input procedure must conform to the standards of the version of COBOL that you are using, either ANSI-74 COBOL standards or ANSI-85 COBOL standards.

An overview of the syntax for an input procedure is as follows:

```
 +<----------------------------------------------------- (1)
 |
 | A                       B
------> <INPUT PROCEDURE ---> <COBOL code for input ----> (2)
         statement>             procedure>

(1) <----------------------------------------------------+
                                                       F|
         C                         E                    | G
(2) --------------------------------> <END PROCEDURE ------>
      |                            |    statement>
      | D                          |
      +--> <WORKING-STORAGE -->+
             specification>
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Use the <INPUT PROCEDURE statement> to name the input procedure. You must name your input procedure. This name is entered into the vocabulary and used to reference the input procedure in report specifications. You can also use the <INPUT PROCEDURE statement> to access data files and DMS II data sets previously defined in the vocabulary specification. When you reference the input procedure name in report specifications, these data structures can be reported.
B	The <COBOL code for input procedure> is the COBOL code for the input procedure. During report generation, control is passed to the input procedure, which must produce a complete logical record before control is returned to the report routines. The <COBOL code for input procedure> may include any of the following COBOL or DMS II constructs: 1. Open/close routines. 2. WORKING-STORAGE areas (77 and 01). 3. File declarations. 4. Sort descriptions.

When you reference the input procedure name in report specifications, items within the WORKING-STORAGE 01 areas are reported.

C Take this path if you do not want to modify any elementary item descriptions in the WORKING-STORAGE SECTION of the <COBOL code for input procedure>. RP3VOC automatically includes the record descriptions in the vocabulary, and assigns default editing pictures to the elementary data items.

D Take this path to modify elementary item descriptions in the WORKING-STORAGE SECTION of the <COBOL code for input procedure>. You can modify names, assign editing pictures, and make condition additions.

E The <END PROCEDURE statement> indicates the end of the specification of the current input procedure.

F Take this path to specify additional input procedures.

G Take this path if you have finished specifying all input procedures.

INPUT PROCEDURE EXAMPLE

The University of Echo Park has all the information about its students in one file. Due to the large amount of data on each student, three records of the file have been allotted to each student.

REPORTER III-generated input routines assume each physical record of a system file corresponds to a logical record; that is, all the information about a student is assumed to be in one physical record. Therefore, the university cannot access all the information in the student file with a generated input routine. The university staff must create their own input procedure.

Note that the following input procedure does not contain OPEN or CLOSE routines. In this case, the generated routines are adequate. If generated OPEN/CLOSE routines do not satisfy your needs, you may need to create OPEN or CLOSE routines for your own input procedure.

The following is an input procedure the university could use to access the three physical records from the student file, and to build one logical record to be used as input for reports:

```
INPUT PROCEDURE ACCESS USES STUDENT.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 SWITCH                PIC 9 COMP VALUE ZERO.
01 INFORMATION-IN-FIRST-RECORD.
   03 FILLER             PIC X(50).
   03 NAME               PIC X(20).
   03 FILLER             PIC X(14).
01 INFORMATION-IN-SECOND-RECORD.
   03 FILLER             PIC X(10).
   03 YEAR-IN-SCHOOL     PIC 9.
   03 FILLER             PIC X(73).
01 INFORMATION-IN-THIRD-RECORD.
   03 STUDENT-ADDRESS    PIC X(50).
   03 FILLER             PIC X(34).

PROCEDURE DIVISION.
INPUT-ROUTINE.
    READ STUDENT AT END
         MOVE 1 TO END-OF-INPUT
         GO TO EXIT-ROUTINE.
    ADD 1 TO SWITCH.
```

```
          IF SWITCH = 3
               MOVE 0 TO SWITCH
               MOVE STUDENT-RECORD TO
                    INFORMATION-IN-THIRD-RECORD
               GO TO EXIT-ROUTINE.
          IF SWITCH = 1
               MOVE STUDENT-RECORD TO
                    INFORMATION-IN-FIRST-RECORD.
          IF SWITCH = 2
               MOVE STUDENT-RECORD TO
                    INFORMATION-IN-SECOND-RECORD.
          GO TO INPUT-ROUTINE.
     END-OF-JOB.
     END PROCEDURE.
```

The input procedure assumes that the file STUDENT was presented to RP3VOC in a previous FILE statement. STUDENT-RECORD is the record description for the student file. Each time INPUT-ROUTINE is called, it returns a logical record. In this case a logical record consists of all information about a student contained in the record areas INFORMATION-IN-FIRST-RECORD, INFORMATION-IN-SECOND-RECORD, and INFORMATION-IN-THIRD-RECORD.

COBOL CODE FOR INPUT PROCEDURE

The COBOL code for input procedure is the main element of an input procedure. It is the actual COBOL source code that accesses the data. The code in the PROCEDURE DIVISION may contain file input processes. These processes obtain and manipulate data before passing the data to report formatting routines. The source code must be written following the appropriate ANSI-standard COBOL syntax.

Some input routines require more than the procedure code to obtain data. The following are some examples of code you might want to include in your input procedure:

1. System files and/or DMSII data sets that are not previously described in the vocabulary specification. In this case OPEN and CLOSE routines must be provided for all system files and/or DMSII data bases.

2. Code that requires temporary storage, i.e., WORKING-STORAGE.

3. User-created OPEN or CLOSE routines for instances in which generated OPEN/CLOSE routines are insufficient. OPEN and CLOSE routines must be provided when the USES clause is not used or if any system files and/or DMSII appear in the COBOL code for input procedure.

Some input routines may require a combination of all the preceding conditions.

RP3VOC gives you flexibility by accepting the COBOL source statements in the form of a skeleton COBOL program. The form is similar to the SOURCE FILE FOLLOWS, described in Section 5, with some extensions. The COBOL source must conform to the appropriate ANSI-standard COBOL syntax. The skeleton takes the form shown in the following example.

```
[any legal COBOL construct preceding FILE-CONTROL]

FILE-CONTROL.
     [SELECT statements] (OPTIONAL)

DATA DIVISION.
FILE SECTION.

     [FD(s) for system files not defined to VOCAL] (OPTIONAL)
WORKING-STORAGE SECTION.
     [ 77(s) ] (OPTIONAL)
     [ 01(s) ]

PROCEDURE DIVISION.

OPEN-ROUTINE.  (OPTIONAL)
     .
     .
     .
[open set code]
[open file code]
     .
     .
     .
    GO TO EXIT-ROUTINE.

CLOSE-ROUTINE.  (OPTIONAL)
     .
     .
     .
[close set code]
[close file code]
     .
     .
     .
    GO TO EXIT-ROUTINE.

INPUT-ROUTINE.
     .
     .
     .
[input accessing code]
     .
     .
     .
    MOVE 1 TO END-OF-INPUT.
    GO TO EXIT-ROUTINE.

END-OF-JOB.
```

RP3VOC requires that the skeletal COBOL program contain the following constructs and paragraph names:

1. FILE-CONTROL.
2. DATA DIVISION.
3. PROCEDURE DIVISION.
4. INPUT-ROUTINE.
5. END-OF-JOB.

The rules of ANSI-74 and ANSI-85 COBOL for column position and specification apply to all COBOL source code except END-OF-JOB, which must begin in column 8, and the COPY statement, which must begin on a separate line. For more information on the COPY statement, see Appendix B, C, or D.

The USING clause of the PROCEDURE DIVISION is ignored for the A Series, and not applicable for the B 2000-B 4000 Series of Systems.

The optional OPEN and CLOSE routines must have the paragraph name and OPEN-ROUTINE and CLOSE-ROUTINE. If the OPEN and/or CLOSE routines are used, all files referenced by the INPUT PROCEDURE must be opened and/or closed.

The input procedure must contain the paragraph name INPUT-ROUTINE. Each routine must return control to the REPORTER III-generated label EXIT-ROUTINE after execution of the routine.

When no more logical records will be passed to the reporting program, the INPUT-ROUTINE must set END-OF-INPUT TO 1. REPORTER III automatically puts END-OF-INPUT in the generated COBOL source. END-OF-INPUT should not appear in the WORKING-STORAGE section of the input procedure.

In addition to referencing item names in system files and/or DMS II structures, the INPUT PROCEDURE can reference data names in the report ACCEPT statement. The examples below show an RP3VOC specification and a REPORTER III specification in which accepted data is used.

RP3VOC Specification:

```
    VOCAB ARE "INPUT"/"V1" AND
    "INPUT"/"V2".

    LIST VOCAB LIB SYNTAX.
```

```
SOURCE FILE FOLLOWS.
FILE-CONTROL.
   SELECT STUDENT ASSIGN TO DISK.
DATA DIVISION.
FD STUDENT
   VALUE OF FILENAME IS STUDENT-FILE-NAME
   RECORD CONTAINS 84 CHARACTERS
   BLOCK CONTAINS 30 RECORDS.

01 STUDENT-RECORD         PIC X(84).
END-OF-JOB.

FILE STUDENT.
END FILE.
INPUT PROCEDURE ACCESS USES STUDENT.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 SWITCH         PIC 9 COMP VALUE ZERO.

01 STUDENT-FILE-NAME          PIC X(6).

01 INFORMATION-IN-FIRST-RECORD.
   03 FILLER                  PIC X(50).
   03 NAME                    PIC X(20).
   03 FILLER                  PIC X(14).
01 INFORMATION-IN-SECOND-RECORD.
   03 FILLER          PIC X(10).
   03 YEAR-IN-SCHOOL  PIC 9.
   03 FILLER          PIC X(73).
01 INFORMATION-IN-THIRD RECORD.
   03 STUDENT-ADDRESS         PIC X(50).
   03 FILLER                  PIC X(34).
PROCEDURE DIVISION.
OPEN-ROUTINE.
   MOVE Q-TODAYS-FILE TO
      STUDENT-FILE-NAME.
   OPEN INPUT STUDENT.
   GO TO EXIT-ROUTINE.

INPUT-ROUTINE.
   READ STUDENT AT END
     MOVE 1 TO END-OF-INPUT
     GO TO EXIT-ROUTINE.
   ADD 1 TO SWITCH.
   IF SWITCH = 3
     MOVE 0 TO SWITCH
     MOVE STUDENT-RECORD TO
          INFORMATION-IN-THIRD-RECORD.
     GO TO EXIT-ROUTINE
   IF SWITCH = 1
     MOVE STUDENT-RECORD TO
          INFORMATION-IN-FIRST-RECORD.
      IF SWITCH = 2
        MOVE STUDENT-RECORD TO
              INFORMATION-IN-SECOND-RECORD.
      GO TO INPUT-ROUTINE.
   END-OF-JOB.
   END PROCEDURE.
```

REPORTER III Reporter Specification:

```
   VOCAB "INPUT"/"V1"

   SUPPRESS COBOL.
   SAVE PARAMETERS AS "INPUT"/"PARAM".
   SAVE COBOL SOURCE AS "INPUT"/"SOURCE".
   SAVE OBJECT AS "INPUT"/"OBJECT".
   ASSIGN ACCEPTED-DATA TO ODT.
   ACCEPT TODAYS-FILE STRING(6) DISPLAY
   "ENTER TODAYS FILE NAME WITH PERIOD(EX.STDNT.)"
   INPUT ACCESS.
   REPORT NAME, STUDENT-ADDRESS, YEAR-IN-SCHOOL.
```

If the accepted data name has less than 22 characters, this name must be prefixed by "Q-" when referenced in the COBOL code for input procedure. If the accepted data name is more than 22 characters, do not use a prefix.

If the 77-items are present in WORKING STORAGE, they must appear before any 01 level. The 77-items do not become entries in the vocabulary but are included in the general report program. All 01s and the associated elementary item <data names> are added to the vocabulary. You can assign editing pictures to elementary items by using the Data-name-change statement (see WORKING-STORAGE specification).

Files presented in the DATA DIVISION must have COBOL SELECT statements following the required FILE-CONTROL construct. These files are considered part of the input procedure, are not added to the vocabulary, and cannot be reported. To report on any data item in a system file and/or DMSII data structure it must be moved to a WORKING-STORAGE data name other than a 77-level item.

The following example shows the skeleton COBOL code in the minimum form:

```
FILE-CONTROL.
DATA DIVISION.
PROCEDURE DIVISION.
INPUT-ROUTINE.
    .
    .
    .
    MOVE 1 TO END-OF-INPUT.
    GO TO EXIT-ROUTINE.
END-OF-JOB.
```

END PROCEDURE STATEMENT

The END PROCEDURE statement indicates the end of the input procedure. RP3VOC adds any remaining WORKING-STORAGE data names to the vocabulary, and then stores the COBOL code in the PROCEDURE DIVISION. RP3VOC is then ready to accept another INPUT statement if present.

Every INPUT statement must have an accompanying END-PROCEDURE statement. The syntax for the END PROCEDURE statement is as follows:

```
                 A
--->END---------->PROCEDURE----------> .
               |                 |
               | B               |
               +--->ROUTINE--->+
```

Path	Explanation
A-B	These paths are equivalent.

Examples:

```
END PROCEDURE.
END ROUTINE.
END PROC.
```

INPUT PROCEDURE STATEMENT

The INPUT PROCEDURE statement instructs RP3VOC that the COBOL code which follows is to be catalogued as an input procedure.

You can also use the USES clause of the input procedure to access previously defined data structures (system files and/or DMS II data sets). The syntax for the INPUT PROCEDURE statement is as follows:

```
                A                  C                     D
--->INPUT--------->PROCEDURE----> <name> ---------------------> (1)
          |                 |              |
          | B               |              | E
          +--->ROUTINE--->+                +--->USES-------> (2)

                                                                    L
(1) -------------------------------------------------------------> .
                                                                 |
      +<--------------------- , <-----------------+              |
      |                                           K|             |
      | F                   G                      |             |
(2) --------> <data-structure -----------------------------------+
                name>         |                        |
                              | H                      |
                              |--->SEQUENTIALLY--->|
                              |                        |
                              | I                      |
                              |--->RANDOMLY------->|
                              |                        |
                              | J                      |
                              +--->DYNAMICALLY---->+
```

The paths of this syntax diagram are described below:

Path	Explanation
A-B	These paths are equivalent.
C	Specify the name of the input procedure here. This name is entered into the vocabulary and may be referenced in the INPUT statement of the report specification.

D Take this path if you do not want the input procedure to access previously defined data structures. Use this path if the COBOL code for input procedure declares all the necessary files and record descriptions, and properly opens and closes these files.

Example:

```
INPUT PROCEDURE TRANS-INFO.
```

E The USES clause allows the input procedure to access previously defined data structures.

Example:

```
INPUT PROCEDURE INPUT-CUST USES CUST-INFO.
```

F The <data-structure name> is the first part of the USES clause. The data-structure name identifies the data structures that you want the input procedure to access. If you include an embedded data set in the USES clause, you must first specify its higher data set in the clause.

G Take this path if the data structure is not a system file or for system files if the input routine accesses the system file sequentially. Sequential access is the default.

H-J These paths only apply if the data structure is a system file. Use these paths to specify if the input routine accesses the system file RANDOMLY, SEQUENTIALLY, or DYNAMICALLY. You can specify DYNAMICALLY only if the system file organization is INDEXED or RELATIVE.

If you specify RANDOMLY with a sequentially organized file, the COBOL SELECT statement generated by the report routines declares a random file access mode, and specifies an ACTUAL KEY clause. The name of the key is in the format QK<file name>. For example, for the random access file PAY-ROLL, the key name is QKPAY-ROLL. A 77-level item is also generated to declare the key. Therefore, the input routine must follow the naming convention when accessing a random file.

Example:

```
INPUT PROC PAYROLL-INPUT USES PAYROLL-FILE
  RANDOMLY.
```

K Take this path if you want the input procedure to access additional data structures.

Example:

```
     INPUT PROCEDURE ACCESS-TRANS-ACCT
       USES ACCT-TRANS, ACCT-FILE RANDOMLY.
```

In this example, two system files are used. ACCT-TRANS is accessed sequentially; ACCT-FILE is accessed randomly.

L Take this path when you have finished specifying all the data structures.

WORKING-STORAGE SPECIFICATION

Use the WORKING-STORAGE specification to modify record elementary item descriptions in the WORKING-STORAGE SECTION of the <COBOL code for input procedure>. The COBOL source must conform to the appropriate ANSI standards.

Except for 77-level items, all COBOL data names pertaining to the declaration of a WORKING-STORAGE record automatically become elements of the vocabulary. By default, RP3VOC assigns editing pictures to each data item based on storage picture.

Since RP3VOC processes the WORKING-STORAGE SECTION sequentially, the modification statements must be specified in the same order as the occurrence of the data names.

The WORKING-STORAGE specification must immediately follow the END-OF-JOB construct. It is not preceded by a FILE statement, or terminated by an END FILE statement.

Example:

```
INPUT PROCEDURE ACCESS USES STUDENT.
DATA DIVISION.
WORKING-STORAGE SECTION.
77 SWITCH                 PIC 9 COMP VALUE ZERO.
01 INFORMATION-IN-FIRST-RECORD.
   03 FILLER              PIC X(50).
   03 NAME                PIC X(20).
   03 FILLER              PIC X(14).
01 INFORMATION-IN-SECOND-RECORD.
   03 FILLER              PIC X(10).
   03 YEAR-IN-SCHOOL      PIC 9.
   03 FILLER              PIC X(73).
01 INFORMATION-IN-THIRD-RECORD.
   03 STUDENT-ADDRESS     PIC X(50).
   03 FILLER              PIC X(34).

PROCEDURE DIVISION.
INPUT-ROUTINE.
    READ STUDENT AT END
         MOVE 1 TO END-OF-INPUT
         GO TO EXIT-ROUTINE.
    ADD 1 TO SWITCH.
    IF SWITCH = 3
```

```
             MOVE 0 TO SWITCH
             MOVE STUDENT-RECORD TO
                  INFORMATION-IN-THIRD-RECORD
             GO TO EXIT-ROUTINE.
        IF SWITCH = 1
             MOVE STUDENT-RECORD TO
                  INFORMATION-IN-FIRST-RECORD.
        IF SWITCH = 2
             MOVE STUDENT-RECORD TO
                  INFORMATION-IN-SECOND-RECORD.
        GO TO INPUT-ROUTINE.
    END-OF-JOB.
    FILLER OF NAME OF INFORMATION-IN-FIRST-RECORD
        IS STUDENT-ID WITH PIC "9(14)".
    END PROCEDURE.
```

The syntax for the WORKING-STORAGE specification is as follows:

```
     +<----------------------------------------------+
     |                                              C|
     |     A                                         | D
-------------> <data-name-change statement> ------------>
        |                                        |
        | B                                      |
        +---> <condition-addition statement> -->+
```

The paths of this syntax diagram are explained below:

Path	Explanation
A	Use the <Data-name-change statement> to assign a new name and/or editing picture to a group or data item in the COBOL description. The name and/or an editing picture is used to report the item. You can also use this statement to effectively delete an item from the description. For details, refer to the Data-name-change statement in Section 5.
B	Use the <Condition-addition statement> to add a condition for a COBOL elementary item in the COBOL description. For details, refer to the Condition-addition statement in Section 5.

C	Take this path to specify additional modifications.
D	Take this path when the <WORKING-STORAGE specification> is complete.

SECTION 7

SYSTEM OPERATIONS ASSOCIATED WITH VOCABULARY PREPARATION

This section contains information on system operations associated with REPORTER III vocabulary preparation on each of the following:

1. A Series of Systems.
2. B 1000 Series of Systems.
3. B 2000/B 3000/B 4000 Series of Systems.

The information is presented separately (under a different main paragraph heading) for each of the three series of systems.

CAUTION

The report generation programs (RP3REP and RP3GEN) should not be run during execution of the vocabulary programs (RP3VOC, RP3VDM, and RP3VGN). Running these programs simultaneously can cause corruption or loss of the new vocabulary files.

A SERIES OPERATIONS

The following subjects are discussed below with respect to use of the REPORTER III System on a computer in the A Series of Systems:

1. The files which must be present on disk before a vocabulary can be prepared.

2. The sequence of statements needed to execute the entire process from input of the vocabulary specifications to vocabulary output.

3. The types of procedures you can use to execute the RP3VDM program:

    a. Automatic execution procedure.
    b. User-controlled execution procedure.
    c. Execution using Work Flow Language (WFL).

FILES REQUIRED FOR EXECUTION

The REPORTER III System consists of the following four files for specifying and generating the vocabulary files. The files are supplied on the REPORTER III System tape and must be present on disk when vocabulary specifications are analyzed and a vocabulary program generated.

1. RP3VOC contains the object code for the REPORTER III Vocabulary Language parsing program.

2. RP3VGM contains the REPORTER III Vocabulary Language syntax definitions used by RP3VOC to parse the vocabulary specifications.

3. RP3VDM contains the object code to build the description of DMS II data bases when required in the vocabulary specifications.

4. RP3VGN contains the object code for the COBOL program generator. The RP3VGN program generates a skeletal program if you specify the syntax option in the LIST statement (see Section 3).

In addition to the above files, other files must be present on disk or disk pack, as follows:

1. The COBOL74 compiler or COBOL85 compiler must be present to compile the generated program. COBOL74 or COBOL85 is used for all applications, including those that access DMS II data structures.

2. If DMS II files are to be accessed, the program DATABASE/INTERFACE as well as the description file DESCRIPTION/<data-base name> must be on disk or disk pack.

   The RP3VOC program also requires that the DB/INVOKE files must be present. The complete file name is DB/INVOKE/<data-base name>, or DB/ INVOKE/<data-base name>/<logical-data-base name>. If the required DB/INVOKE file is not present, RP3VOC will automatically run RP3VDM to create it.

INPUT REQUIRED FOR EXECUTION

The process of REPORTER III vocabulary specification analysis and vocabulary production is automatic. You need input only one simple sequence of statements to execute the entire process from specifications input to vocabulary output.

The Work Flow Language (WFL) input execution deck consists of the following components, in the order presented:

1. RUN statement
2. FILE statement
3. DATA statement
4. Vocabulary specifications file
5. ?END statement

Descriptions of each of these five components follow. For those users using card input, the RUN, FILE, and DATA statements must be preceeded by an invalid punch.

RUN Statement

The RUN statement instructs the Master Control Program (MCP) to schedule the RP3VOC program for execution of specification parsing. The syntax is as follows:

```
    RUN RP3VOC
```

FILE Statement(s)

The FILE statement is optional. It allows you to change the external file name and/or hardware device for one or more of the files used by RP3VOC. Any number of FILE statements can be included. (Refer to the A Series Work Flow Language Reference Manual for the syntax of the FILE statement.)

Each file used has two names associated with it.

1. The internal file name is the name by which a program refers to the file.

2. The external file name is the name by which the MCP refers to the file.

The two names can be the same, but this is not a requirement. To change file names at execution time, you need to "equate" the internal file name to the new external file name through use of a FILE statement. A hardware device different from the one normally used to contain the file can be specified together with the new external file name.

To use a FILE statement, you must know the internal file name of each file that is to be equated. Table 7-1 gives the internal file names, the default values for the external file names, the hardware device, and a description of each file used by the analysis program, RP3VOC.

You can use the following FILE statement to change the name of the RP3VCR input file:

```
FILE RP3VCR (TITLE = MYSPECS, KIND = DISK)
```

Table 7-1

RP3VOC File Equates for A Series of Systems

Internal File Name	Default External File Name & Kind	Description
RP3VGM	RP3VGM, disk	Grammar definition file.
RP3VCR	RP3VCR, card reader	Vocabulary specifications input from card reader.
RP3VPR	RP3VPR, printer	File where the vocabulary specifications are printed (and error messages listed).

DATA Statement

The DATA statement is placed immediately before the vocabulary specifications for card decks or WFL decks (the statement is not used if input of the vocabulary specifications is from disk). The two syntax formats for the statement are as follows:

1. DATA RP3VCR
2. EBCDIC RP3VCR

NOTE

RP3VOC does not accept BCL-coded input.

Vocabulary Specifications File

The vocabulary specifications file consists of all the language statements which you have constructed in designing your vocabulary. This file is placed between the DATA statement and the END statement. If the vocabulary specifications are a disk file, the DATA and END cards are not used. The file must be structured similarly to files containing COBOL source code. The sequence numbers may appear in columns 1 through 6, column 7 is blank, and the report specifications appear in columns 8 through 72.

?END Statement

The ?END statement, which signals the end of your input, must be the last statement in the input execution deck. The syntax is as follows:

```
?END
```

EXECUTION PROCEDURE

Before you can process your vocabulary specifications, the REPORTER III System programs and files must be loaded to disk. The COBOL74 compiler or COBOL85 compiler must be accessible for processing. If a DMS II data base is used, the DATABASE/INTERFACE file and the DESCRIPTION/<data-base name> file must be accessible for processing.

The REPORTER III System has the capability to proceed automatically from specifications input to vocabulary output. You can also use the Work Flow Language (WFL) in connection with REPORTER III on the A Series of Systems. These two capabilities are discussed below.

Automatic Execution Procedure

The procedure for the default automatic execution of the REPORTER III vocabulary preparation process is as follows:

1. You input the execution deck (from the card reader or terminal, whichever applies) to begin execution of RP3VOC. The RP3VOC execution can result in the creation of two disk files: a vocabulary library file, which contains the COBOL source code extracted from specified source programs, and a file containing a vocabulary of names.

   RP3VOC is the first phase in the vocabulary preparation process. It parses the vocabulary specifications and produces the vocabulary files used by RP3REP. If you specify a syntax compile, RP3VOC will automatically run RP3VGN.

2. If DB/INVOKE files are not available for data bases described in the vocabulary specification, RP3VOC automatically executes RP3VDM.

3. If you specify the syntax option of the LIST statement, a syntax compile is performed on the vocabulary. In this case, once RP3VOC is terminated and no errors occur in the vocabulary specification, RP3VOC automatically executes RP3VGN.

   RP3VGN is run to generate a source program based on the vocabulary files produced by RP3VOC.

   When RP3VGN terminates, the COBOL source program is passed to the COBOL74 compiler or COBOL85 compiler automatically for a SYNTAX compile. |

4. If you do not specify a syntax compile, the execution is complete when RP3VOC terminates.

<u>User-Controlled Execution Procedure</u>

You also have the option to run RP3VDM stand-alone. Program RP3VDM creates a file called DB/INVOKE/<data-base name> or DB-INVOKE/<data-base name>/<logical-data-base name>. This file is a directory of the A Series DMS II data base or logical data base used by RP3VOC.

If RP3VDM is to be run stand-alone, it must be run once for every DMS II data base whose description is added to the vocabulary through RP3VOC. Once the DB-INVOKE file has been created, it need only be recreated if the DASDL description of the data base is altered.

Required Files:

The following files must be present to run RP3VDM:

1. COBOL74 COMPILER or COBOL85 COMPILER.
2. DATABASE/INTERFACE.
3. DESCRIPTION/<data-base name>.

Running RP3VDM:

Input the <data-base name> in the following manner:

```
----->RUN RP3VDM (" DB--------------------------------------> (1)

(1)------> <data-base name> ---------------------------------> (2)
       |                                                   |
       +--> <logical-data-base name> OF <data-base name> -->+

(2)------------------------------------------------------> ") -->
       |                         |  |               |
       +--> ,LIST-----------------|  |--> ,COBOL74-->|
                   |             |  |               |
                   +--> ,SYNTAX---->+  +--> ,COBOL85-->+
```

The <data-base name> must be a valid A Series DMS II data base. You must reference data base descriptions in sequential order when specifying the data base to RP3VCC.

In addition, RP3VDM offers you the following options:

1. The LIST option provides a listing of the DB/INVOKE file created by RP3VDM. Use this listing to see if all data elements from the data base have been included in the DB/INVOKE file.

2. The SYNTAX option prevents the DB/INVOKE file from being saved. RP3VDM compiles with the COBOL74 compiler or COBOL85 compiler to check the syntax of the data base without saving the DB/INVOKE file.

3. The COBOL74 option selects the COBOL74 compiler. This is the default option. If you do not designate either the COBOL74 or the COBOL85 option, the COBOL74 compiler is used.

4. The COBOL85 option selects the COBOL85 compiler.

Examples:

```
     RUN RP3VDM ("DB UNIV")
     RUN RP3VDM ("DB UNIV, LIST")
     RUN RP3VDM ("DB UNIV, LIST, SYNTAX")
     RUN RP3VDM ("DB UNIVERSITY OF UNIV, LIST")
     RUN RP3VDM ("DB UNIV, LIST, COBOL85")
```

Execution Using Work Flow Language (WFL)

A Work Flow Language capability can be used in connection with the REPORTER III System on the A Series of Systems to facilitate user control of the REPORTER III System, and take maximum advantage of the MCP. The use of WFL is optional.

The following is an example of a WFL execution deck for vocabulary production:

```
     BEGIN JOB VOCAB/SYSTEM/WFL; %SPECIFIC WFL EXAMPLE
     USER = QQSV123/PASSWORD
     CHARGE = AR/DEVELOPEMNT/095;
     PRIORITY = 55;
     RUN RP3VOC;
     DATA RP3VCR
        VOCAB FILE ARE "VOCFILE1" AND "VOCFILE2".
              :
              :
              :
     ?END
```

Additional WFL statements may precede the RUN statement depending on your system's individual requirements.

RUNNING THROUGH THE ODT

The following is an example of running RP3VOC through the Operator Display Terminal (ODT). It assumes the vocabulary specifications are in a disk file called NEW/VOC/SPECS:

```
     RUN RP3VOC; FILE RP3VCR
     (TITLE=NEW/VOC/SPECS, KIND=DISK).
```

B 1000 OPERATIONS

The following subjects are discussed below with respect to use of the REPORTER III System on a computer in the B 1000 Series of Systems:

1. The files which must be present on disk before a vocabulary can be prepared.

2. The sequence of statements needed to execute the entire process from input of the vocabulary specifications to vocabulary output.

FILES REQUIRED FOR EXECUTION

The REPORTER III System consists of the following three files for specifying and generating the vocabulary files. The files are supplied on the REPORTER III System tape and must be present on disk when vocabulary specifications are analyzed and a vocabulary program generated.

1. RP3VOC contains the object code for the REPORTER III Vocabulary Language parsing program.

2. RP3VGM contains the REPORTER III Vocabulary Language syntax definitions used by RP3VOC to parse the vocabulary specifications.

3. RP3VGN contains the object code for the COBOL program generator. The RP3VGN program generates a skeletal program if you specify the syntax option in the LIST statement (see Section 3).

In addition to the above files, other files must be present on disk or disk pack:

1. The COBOL74 compiler must be present to compile the generated program. The COBOL74 interpreter is required for RP3VGN.

2. If DMS II files are to be accessed, the following dictionary and library files created by DASDL must be present on disk or disk pack:

   a. <data-base name>/DICTIONARY
   b. #<data-base name>/disjoint-data-set name>

INPUT REQUIRED FOR EXECUTION

The process of REPORTER III vocabulary specification analysis and vocabulary production is automatic. You need input only one simple sequence of statements to execute the entire process from specifications input to vocabulary output. The input execution deck consists of the following components, in the order presented:

1. ?EXECUTE statement
2. ?FILE statement
3. ?DATA statement
4. Vocabulary specifications file
5. ?END statement

Descriptions of each of these five components follow.

?EXECUTE Statement

The ?EXECUTE statement instructs the Master Control Program (MCP) to schedule the RP3VOC program for execution of specification parsing. The syntax is as follows:

```
?EXECUTE RP3VOC
```

?FILE Statement(s)

The ?FILE statement is optional. It allows you to change the external file ID and/or hardware device for one or more of the files used by RP3VOC. Any number of ?FILE statements can be included. The syntax is as follows:

```
?FILE internal file name NAME external file ID
   <output device>;
```

Each file used has two names associated with it.

1. The internal file name is the name by which a program refers to the file.

2. The external file ID is the name by which the MCP refers to the file.

The two names can be the same, but this is not a requirement. To change file names at execution time, you need to "equate" the internal file name to the new external file ID through use of a ?FILE statement. A hardware device different than the one normally used for a file can be specified together with the new external file ID.

To use a ?FILE statement, you must know the internal file name of each file that is to be equated. Table 7-2 gives the internal file names, the default values for the external file IDs, the hardware device, and a description of each file used by the analysis program, RP3VOC.

Table 7-2

RP3VOC File Equates for B 1000
Series of Systems

Internal File Name	Default External File Name & Kind	Description
RP3VGM	RP3VGM, disk	Grammar definition file.
RP3VCR	RP3VCR, card reader	Vocabulary specifications input from card reader.
RP3VPR	RP3VPR, printer	File where the vocabulary specifications are printed (and error messages listed).

NOTE

If a disk file is equated to RP3VCR, use the word DEFAULT or DEF after the external file ID. DEFAULT matches the block and record size of the disk file to RP3VCR.

Example:

The name of the RP3GRM file is to be changed to IMS001. The ?FILE statement for the change is as follows:

```
    ?FILE RP3GRM NAME IMS001;
```

It is possible to change the hardware medium for one or more files. For example, if you want the output file RP3PRT to be sent to printer backup disk without changing the name of the file, the ?FILE statement for the change in output medium is as follows:

```
?FILE RP3PRT NAME RP3PRT BACKUP.DISK NO HARDWARE;
```

If the name of the input file is to be changed, for example, to PAY-ROLL/REPORT/EMPLOYEES and the input medium is to be changed to disk cartridge, the ?FILE statement for the change is as follows:

```
?FILE RP3CRD NAME PAY-ROLL/REPORT/EMPLOYEES
   DISK.CARTRIDGE DEF;
```

For further information, refer to the B 1000 Systems Software Operational Guide.

?DATA Statement

The ?DATA statement names the vocabulary specifications for the MCP and tells the computer the character set used for the input execution deck. The statement is placed immediately before the vocabulary specifications for card decks or pseudo-reader decks. The DATA card is not used in vocabulary specification files which are disk files. If you use card input, only cards punched in EBCDIC can be input to RP3VOC.

The syntax for the statement is as follows:

```
?DATA RP3VCR
```

Vocabulary Specifications File

The vocabulary specifications file consists of all the language statements which you have constructed in designing your vocabulary. This file is placed between the ?DATA statement and the ?END statement. If the vocabulary specifications are a disk file, the DATA and END cards are not used. The file must be structured similarly to the file containing COBOL source code. The sequence numbers may appear in columns 1 through 6, column 7 is blank, and the report specifications appear in columns 8 through 72.

?END Statement

The ?END statement, which signals the end of your input, must be the last statement in the input execution deck. The syntax is as follows:

```
?END
```

EXECUTION PROCEDURE

Before you can process your vocabulary specifications, the REPORTER III System programs and files must be loaded to disk or disk pack. The COBOL74 compiler must be accessible for processing. If a DMS II data base is to be used, the DMS II dictionary and library files must be accessible.

The REPORTER III System has the capability to proceed automatically from specifications input to vocabulary output. This capability is discussed below.

Automatic Execution Procedure

The procedure for the default automatic execution of the REPORTER III vocabulary preparation process is as follows:

1. You input the execution deck (from the card reader or terminal, whichever applies) to begin execution of RP3VOC. The following message is displayed:

   ```
   RP3VOC = <mix#> BOJ
   ```

The RP3VOC execution can result in the creation of two disk files: a vocabulary library file, which contains the COBOL source code extracted from specified source programs, and a file containing a vocabulary of names.

RP3VOC is the first phase in the vocabulary preparation process. It parses the vocabulary specifications and produces the vocabulary files used by RP3REP. If you specify a syntax compile, RP3VOC executes RP3VGN.

2. If you specify the syntax option of the LIST statement, a syntax compile is performed on the vocabulary. In this case, once RP3VOC is terminated and no errors occur in the vocabulary specification, RP3VOC automatically executes RP3VGN. The following message is displayed:

   ```
   RP3VGN = <mix#> BOJ.
   RP3VOC = <mix#> EOJ.
   ```

   RP3VGN is run to generate a source program based on the vocabulary files produced by RP3VOC.

   When RP3VGN terminates, the COBOL source program is passed to the COBOL74 compiler automatically. The following message is displayed:

   ```
   COBOL74: COBOL-program = <mix#> BOJ.
   RP3VGN = <mix#> EOJ.
   ```

   At the end of compilation, the following message is produced:

   ```
   COBOL74: COBOL-program = <mix#> EOJ.
   ```

3. If you do not specify a syntax compile, the execution is complete when RP3VOC terminates. The following message is displayed:

   ```
   RP3VOC= <mix> EOJ.
   ```

EXECUTION THROUGH THE ODT

The following is an example of running RP3VOC through the Operator Display Terminal (ODT). It assumes the vocabulary specifications are in a disk file called NEW.VOC/SPECS:

```
RP3VOC; FILE RP3VCR NAME NEW.VOC/SPECS DISK DEF.
```

B 2000/B 3000/B 4000 OPERATIONS

The following subjects are discussed below with respect to use of the REPORTER III System on a computer in the B 2000/B 3000/B 4000 Series of Systems:

1. The files which must be present on disk before a vocabulary can be prepared.

2. The sequence of statements needed to execute the entire process from input of the vocabulary specifications to vocabulary output.

3. RP3VDM execution procedure.

FILES REQUIRED FOR EXECUTION

The REPORTER III System consists of the following five files for specifying and generating the vocabulary files. The files are supplied on the REPORTER III System tape and must be present on disk when vocabulary specifications are analyzed and a vocabulary program generated.

1. RP3VOC contains the object code for the REPORTER III Vocabulary Language parsing program.

2. RP3VGM contains the REPORTER III Vocabulary Language syntax definitions used by RP3VOC to parse the vocabulary specifications.

3. RP3VDM contains the object code to build the description of DMS II data bases when required in the vocabulary specifications.

4. RP3VGN contains the object code for the COBOL program generator. The RP3VGN program generates a skeletal program if you specify the syntax option in the LIST statement (see Section 3).

In addition to the above files, other files must be present on disk or disk pack, as follows:

1. The COBOL compiler must be present to compile the generated program. COBOL is used for all applications.

2. If DMS II files are to be accessed, the files created by RP3VDM must be available. Refer to the section describing RP3VDM operations for the naming conventions for these files.

INPUT REQUIRED FOR EXECUTION

The process of REPORTER III vocabulary specification analysis and vocabulary production is automatic. You need input only one simple sequence of statements to execute the entire process from specifications input to vocabulary output. The input execution deck consists of the following components, in the order presented:

1. ?EXECUTE statement
2. ?FILE statement
3. ?DATA statement
4. Vocabulary specifications file
5. ?END statement

Descriptions of each of these five components follow.

?EXECUTE Statement

The ?EXECUTE statement instructs the Master Control Program (MCP) to schedule the RP3VOC program for execution of specification parsing. The following are examples of executing RP3VOC using different media files.

1. For card or pseudo-reader files:

```
?EXECUTE RP3VOC
?DATA RP3VCR
  :
  :
Vocabulary specification
  :
  :
?END
```

2. For disk files (assuming the vocabulary file name is VOCSPC):

   ```
   ?EXECUTE RP3VOC VA 0 3 FILE RP3VDK VOCSPC
   ```

   In this case, enter the command on the Operator Display Terminal (ODT). VOCSPC resides on disk as a DATA or COBOL type file

3. For disk pack files (assuming the vocabulary file name is VOCSPC):

   ```
   ?EXECUTE RP3VOC VA 0 3 FILE RP3VDK
      REPORT/VOCSPC DPK
   ```

   In this case, enter this command through an ODT. VOCSPC resides on pack REPORT as a DATA or COBOL type file.

?FILE Statement(s)

The ?FILE statement is optional. It allows you to change the external file ID and/or hardware device for one or more of the files used by RP3VOC. Any number of FILE statements can be included.

Each file used has two names associated with it.

1. The internal file name is the name by which a program refers to the file.

2. The external file ID is the name by which the MCP refers to the file.

The two names can be the same, but this is not a requirement. To change file names at execution time, you need to "equate" the internal file name to the new external file ID through use of a FILE statement. A backup output device other than the one normally used for a file can be specified together with the new external file ID.

To use a ?FILE statement, you must know the internal file name of each file that is to be equated. Table 7-3 gives the internal file names, the default values for the external file IDs, the hardware device, and a description of each file used by the analysis program, RP3VOC.

Table 7-3

RP3VOC File Equates for B 2000/
B 3000/B 4000 Series of Systems

Internal File Name	Default External File Name & Kind	Description
RP3VGM	RP3VGM, disk	Grammar definition file.
RP3VCR	RP3VCR, card reader	Vocabulary specifications input from card reader.
RP3VPR	RP3VPR, printer	File where the vocabulary specifications are printed (and error messages listed).
RP3VDK	RP3VDK, disk	Vocabulary specifications input from disk. Use VA 0=3 in the EXECUTE statement to specify disk input.

Example:

The name of the RP3GRM file is to be changed to IMS001. The ?FILE statement for the change is as follows:

```
?FILE RP3GRM = IMS001
```

It is possible to specify that a backup file be used instead of the physical hardware device. For example, if you want the output file RP3PRT to be sent to printer backup disk without changing the name of the file, the ?FILE statement for the change in the output medium is as follows:

```
?FILE RP3PRT = RP3PRT PRINT DISK
```

For further information, refer to the B 2000/B 3000/B 4000 Systems Software Operational Guide.

?DATA Statement

The ?DATA statement names the vocabulary specifications for the MCP and tells the computer the character set used for the input execution deck. The statement is placed immediately before the vocabulary specifications for card decks or pseudo-reader decks. The DATA card is not used in vocabulary specification files which are disk files. If you use card input, only cards punched in EBCDIC can be input to RP3VOC.

The syntax for the ?DATA statement is as follows:

```
?DATA RP3VCR
```

Vocabulary Specifications File

The vocabulary specifications consist of all the language statements which you have constructed in designing your vocabulary. The specifications are placed between the ?DATA statement and the ?END statement. If the vocabulary specifications are a disk file, the DATA and END card are not used. The file must be structured similarly to a file containing COBOL source code. The sequence numbers may appear in columns 1 through 6, column 7 is blank, and the report specifications appear in column 8 through 72.

?END Statement

The ?END statement, which signals the end of your input, must be the last statement in the input execution deck. The syntax is as follows:

```
?END
```

EXECUTION PROCEDURE

Before you can process your vocabulary specifications, the REPORTER III System programs and files must be loaded to disk. Note that the COBOL compiler must be accessible for processing.

The REPORTER III System has the capability to proceed automatically from specifications input to vocabulary output. This capability is discussed below.

Automatic Execution Procedure

The procedure for the default automatic execution of the REPORTER III vocabulary preparation process is as follows:

1. You input the execution deck (from the card reader or terminal, whichever applies) to begin execution of RP3VOC. The following message is displayed on the ODT:

   ```
   BOJ RP3VOC
   ```

   The RP3VOC execution results in the creation of two disk files: a vocabulary library file, which contains the COBOL source code extracted from specified source programs, and a file containing a vocabulary of names.

   RP3VOC is the first phase in the vocabulary preparation process. It parses the vocabulary specifications and produces the vocabulary files used by RP3REP. If you specify a syntax compile, RP3VOC executes RP3VGN.

2. If you specify the syntax option of the LIST statement, a syntax compile is performed on the vocabulary. In this case, once RP3VOC is terminated and no errors occur in the vocabulary specification, RP3VOC automatically executes RP3VGN. The following message is displayed:

   ```
   BOJ RP3VGN
   EOJ RP3VOC
   ```

   RP3VGN is run to generate a source program based on the vocabulary files produced by RP3VOC.

   When RP3VGN terminates, the COBOL source program is passed to the COBOL compiler automatically.

3. If you do not specify a syntax compile, the execution is complete when RP3VOC terminates.

RP3VDM

The RP3VDM program creates DB-INVOKE files for RP3VOC. The DB-INVOKE file is an 80-byte/record, 9-record/block sequential file and contains a list of all the elements in the specified logical or physical data base.

RP3VDM creates DB-INVOKE files by invoking the logical or physical data base in a dummy COBOL program. RP3VDM then processes the compiler listing to extract the data base description. RP3VOC can then read the DB-INVOKE files and build the vocabulary files. The REPORTER III System uses the the vocabulary files to generate a report program that accesses the data base.

You must execute RP3VDM once for every B 2000/B 3000/B 4000 system DMS II logical or physical data base that RP3VOC uses. Once the DB-INVOKE file is created, it need be recreated only if the DASDL description of the data base is altered (if logical data bases, remaps, data sets, sets, or data items are added or deleted from the DASDL).

The following files must be present for RP3VDM to run:

1. COBOL (ANSI-74) compiler.
2. The DDF file. The DDF file is created when the DASDL description is compiled. If the DDF file is on a restricted disk pack, you specify the pack name to RP3VDM.

RP3VDM-INPUT SPEC

Use the RP3VDM-INPUT SPEC to provide three pieces of information to RP3VOC:

1. The logical- and/or physical-data-base name.
2. Run-time options.
3. The DB-INVOKE file name.

The RP3VDM-INPUT SPEC is given in free form format, appearing in columns 1 through 72 (8 through 72 for COBOL type disk and pack files), and can be on multiple input records.

You can enter the RP3VDM-INPUT SPEC using the following media.

1. The card reader or pseudo-reader.

   Example:

   ```
   ?EX RP3VDM
   ?DATA CARD
   DB LDB1 OF DASD LIST
   ?END
   ```

2. A disk file.

   Example:

   ```
   ?EX RP3VDM VALUE 0 = 1
   ?FILE RP3DDK = INSPC
   ```

3. A pack file.

   Example:

   ```
   ?EX RP3VDM VA 0 1
   ?FILE RP3DDK SYSTEM/INSPC DPK
   ```

4. The Operator Display Terminal (ODT).

   Example:

   ```
   ?EX RP3VDM VA 0 10
   <mix#>AX DB LDB1 OF DASD
   <mix#>AX "LIST DB-INVOKE LOGDB2"
   <mix#>AX END
   ```

   The second accepted input line requires quotes because of the hyphen in DB-INVOKE.

   If you enter a RP3VDM-INPUT SPEC with the ODT, the last entry must be END.

The syntax for the RP3VDM-INPUT SPEC is as follows:

```
            A
---->DB-------------------------------------------> (1)
       |                                  |
       | B                                |
       +--> <logical-data-base name> --->OF-->+
```

```
                                            C
(1) --> <physical-data-base name> ---------------------> (2)
                                       |                 |
                                       |             G   |
                                       |<----- , <---|
                                       |                 |
                                       | D               |
                                       |-->LIST----->|
                                       |                 |
                                       | E               |
                                       |-->SYNTAX--->|
                                       |                 |
                                       | F               |
                                       +-->DEBUG---->+

      H
(2) --------------------------------------------------->
     |                                                 |
     |                                               N |
     |<------------------------------------------------|
     |                                                 |
     | I                                               |
     |--->DDF-PACK-------------> <pack ID> --------->|
     |                                                 |
     | J               K                               |
     +--->DB-INVOKE------> <file name> ------------>|
                        |                              |
                        | L                            |
                        |---> <pack ID>/<file name> --->|
                        |                              |
                        | M                            |
                        +---> <pack ID>/ ------------->+
```

The paths of this syntax diagram are described below:

Path	Explanation
A	Take this path to describe a physical data base. RP3VDM uses the <physical-data-base name> to create a COBOL data base invoke statement in a dummy COBOL program. The compiler produces a printer backup listing that RP3VDM processes and saves as the DB-INVOKE file. The default name of the DB-INVOKE file is the first three letters of the <physical-data-base name> followed by VOC. To specify a non-default name, use path J in the syntax diagram.

B Take this path to describe a logical data base. RP3VDM uses the <logical-> and <physical-data-base name>s to create a COBOL data base invoke statement in a dummy COBOL program. The compiler produces a printer backup listing that RP3VDM processes and saves as the DB-INVOKE file.

You must use path J and K or path L to specify a <file name> for the DB-INVOKE file. Be sure that you specify a different name for each DB-INVOKE file.

C Take this path when you want to use default run-time options. The defaults are as follows:

1. No listing of the DB-INVOKE file.
2. The DB-INVOKE file is saved.
3. No debug listing is produced.

D Take this path if you want a listing of the DB-INVOKE file. You can get the same result from a simple DMPALL listing of the DB-INVOKE file after RP3VDM has been run.

E Take this path if you want a SYNTAX check of the RP3VDM run without producing the DB-INVOKE file.

F Take this path if you want debugging information. This option can also be set at run-time by setting switch 3 to 1.

Example:

```
?EX RP3VDM VALUE 0=100
```

G Take this path to specify additional run-time options.

H Take this path if you want to use default file attributes. You can only use this path if you have not specified a <logical-data-base name> using path B. The default file attributes are:

1. The DDF file is not on a restricted disk pack (the compiler needs no file equate to find the DDF file).

2. The DB-INVOKE file is saved as xxxVOC, where xxx are the first three characters of the <physical-data-base name>. The DB-INVOKE file is an 80-byte/record, 9-record/block sequential file.

I Take this path if the DDF file is on a restricted disk pack. The <pack ID> indicates the location of the disk pack containing the DDF file.

J Take this path to specify a non-default name for the DB-INVOKE file. If you specified a <logical-data-base name> in path C, you must use this path or path M.

K Take this path to assign the <file name> to the DB-INVOKE file. Any <file name> exceeding six characters is truncated to the first six characters. A <file name> must not appear within quotes. The default device is DISK.

Examples:

```
DB LDB1 OF DASD
DB-INVOKE DBLDB1
```

L Take this path to assign the DB-INVOKE file to a <pack ID> under a specified <file name>. The device will be PACK. The <file name> and <pack ID> must not be within quotes.

Examples:

```
DB LDB1 OF DASD
DB-INVOKE DBLDB1/TROPER
```

M Take this path to assign the DB-INVOKE file to a <pack ID> under the default <file name> of xxxVOC, where xxx is the first three characters of the data-base name. The device will be PACK. If you used path B, you cannot use this path.

N Take this path to assign additional non-default attributes through paths I or J.

APPENDIX A

RP3VOC SYSTEM FLOW

Input to the RP3VOC program consists of vocabulary specification statements. These statement may be used to obtain file information from one or more COBOL source programs, A Series and B 2000 - B 4000 Series DMS II Directory files (DB INVOKE files), and/or B 1000 Series DMS II data set COBOL library files. COBOL source programs used as input to RP3VOC may reside on disk, tape, card, disk pack, or library files. A Series and B 2000 - B 4000 DMS II DB INVOKE files, data base descriptions files, and B 1000 Series DMS II data set COBOL library files may reside on disk or disk pack. A schematic representation of the RP3VOC System is illustrated in Figure A-1.

RP3VOC produces two disk files which the RP3REP program uses to generate report requirements. The vocabulary file is a directory of names and their corresponding characteristics, such as COBOL pictures and levels. The other file, a COBOL library file, consists of COBOL source code describing the various files, records, and <input procedure>s specified for RP3VOC. When a REPORTER Language INPUT statement specifies input file(s), the proper data-file descriptions and FDs are obtained from this second file. The output listings of RP3VOC include:

1. A listing of the RP3VOC specification cards used as input to the RP3VOC System.

2. The contents of the two disk files.

RP3VOC recognizes only one data definition per line of source code.

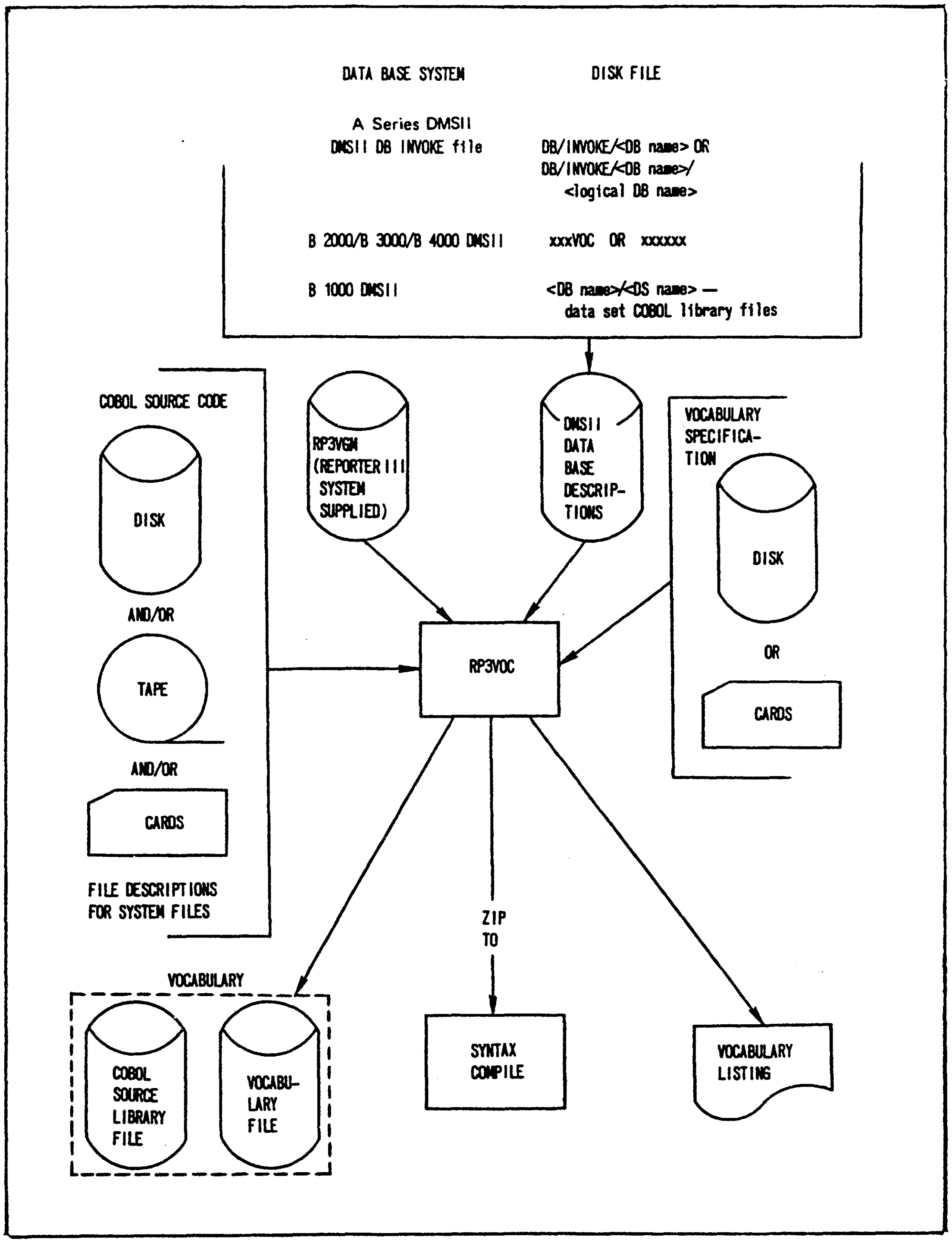

Figure A-1. Schematic Representation of the RP3VOC System

APPENDIX B

UNRECOGNIZED COBOL CONSTRUCTS A SERIES OF SYSTEMS

The RP3VOC portion of the REPORTER III System analyzes user-supplied ANSI-standard COBOL source code, which provides data definitions for elements being included in a vocabulary.

RP3VOC accepts all COBOL constructs as defined in the A Series COBOL reference manuals, with the exception of the continuation indicator. However, some of the COBOL constructs are ignored by RP3VOC. Below is a synopsis of COBOL language structure (as defined in the reference manual), with comments indicating whether the constructs are:

1. Recognized by RP3VOC.
2. Not recognized by RP3VOC.
3. Not applicable to data definition.
4. Accepted when defining an <input procedure>.

COBOL LANGUAGE CONSTRUCTS

RP3VOC recognizes only one data definition per line of source code. The following are the COBOL language constructs recognized by RP3VOC:

1. IDENTIFICATION DIVISION

   This division is not applicable to data definition.

2. ENVIRONMENT DIVISION

   a. CONFIGURATION SECTION

      This section is not applicable to data definition.

b. INPUT-OUTPUT SECTION

1) FILE-CONTROL

The entire SELECT statement is accepted in defining an <input procedure>. RP3VOC recognizes the following SELECT clauses:

a) ASSIGN

The hardware device name in the ASSIGN clause may not exceed 12 characters, and may not contain optional sub-clauses or more than one word.

b) COPY

The COPY statement is recognized by RP3VOC, including FROM...THRU; however, the REPLACING clause is ignored by RP3VOC. The COPY statement must begin on a new line.

The following are examples of COPY statements accepted by RP3VOC.

```
COPY "DEV/LIBRARY01".
```

```
COPY "DEV/LIBRARY02" ON "DEVPACK" FROM 100
THROUGH 250.
```

In the following example, RP3VOC ignores the REPLACING clause of the COPY statement.

```
COPY "LIB01" FROM 500 THRU 600 REPLACING NAME BY
WS-NAME.
```

In the next example, an error will result because the COPY statement does not begin on a separate line.

```
FD INV-FILE COPY "LIB02".
```

c) ORGANIZATION

d) RECORD KEY

Any qualifiers given for the RECORD KEY are ignored. The data item referenced as the key must be unique within the record description entries for the file.

RP3VOC recognizes, but does not process the following clauses:

a) ACCESS MODE
b) FILE STATUS

All other clauses are either not recognized by RP3VOC or are not applicable to data definition.

2) I-O-CONTROL

I-O-CONTROL is not recognized by RP3VOC.

3. DATA DIVISION

a. FILE SECTION

1) FILE DESCRIPTION

RP3VOC recognizes the entire FD and all clauses except the REPLACING clause of the COPY statement. Refer to the COPY statement under FILE-CONTROL for more information. SD is not applicable to data definition. All options under FILE DESCRIPTION are accepted in defining an <input procedure>.

The "VALUE OF TITLE IS <data name>" clause is accepted in file descriptions supplied to RP3VOC. You must supply the TITLE value in the specifications using either: 1) the WITH ID clause of the FILE statement, or 2) an <input procedure>.

2) RECORD DESCRIPTION

Except for the REPLACING...BY clause of the COPY statement, all COPY clauses are recognized by RP3VOC. However, some may not be applicable to data definition. When used, the COPY statement must begin on a new line. For a discussion of additional limitations, refer to the COPY statement under FILE-CONTROL.

The 66 RENAMES is not recognized by RP3VOC. The entire 88 condition name is recognized by RP3VOC.

b. DATA-BASE SECTION

This section is not applicable to data definition. It is accepted by RP3VOC when defining an <input procedure>.

c. WORKING-STORAGE SECTION

1) 77-LEVEL ITEMS

The 77-LEVEL ITEMS are accepted by RP3VOC within the <input procedure>, but are not reportable within report specifications.

2) 01-LEVEL ITEMS

The 01-LEVEL ITEMS are recognized by RP3VOC for <input procedure>s only and have the same limitations described under the heading RECORD DESCRIPTION. For a discussion of additional limitations, refer to the COPY statement under FILE-CONTROL. DEBUG lines are excluded and a warning is given.

d. COMMUNICATION SECTION

This section is not applicable to data definition.

e. REPORT SECTION

This section is not applicable to data definition.

4. PROCEDURE DIVISION

This division is not applicable to data definition but is accepted by RP3VOC when defining an <input procedure>. Refer to the COPY statement under FILE-CONTROL for a discussion of additional limitations. RP3VOC ignores the USING clause of the PROCEDURE DIVISION. If present, DEBUG lines are ignored and a warning is given.

APPENDIX C

UNRECOGNIZED COBOL CONSTRUCTS B 2000/B 3000/B 4000 SERIES OF SYSTEMS

The RP3VOC portion of the REPORTER III System can analyze user-supplied ANSI-74 COBOL source code. This code provides data definitions for elements being included in a vocabulary.

Except for card input, all COBOL source code must be blocked 9 (see Table 5-3). Check source and library files for proper blocking by using the ODT KA command for files on disk, and the PA command for files on disk pack. Refer to MCPVI Keyboard Messages in the B 2000/B 3000/B 4000 Series MCPVI System Software Operations Guide for detailed information.

The blocking check should show each file to be 160 digits (80 characters), blocked 9. If a file is not blocked 9, you can use DMPALL to reblock it. The following two examples show the way you reblock a file on disk or on disk pack using DMPALL.

Examples:

If a KA listing shows a disk file named PROG1 to be blocked 5, use an ODT command to execute DMPALL to change the blocking, as shown:

```
PFM DSKDSK PROG1 80 5 PROG1 80 9
```

If a PA listing shows the PROG1 file blocked 5 on a disk pack named TROPER, the following ODT command would be used to execute DMPALL:

```
PFM DPKDPK TROPER/PROG1 80 5
        TROPER/PROG1 80 9
```

For more information regarding DMPALL, refer to the B 2000/B 3000/B 4000 Series MCPVI System Software Operations Guide.

RP3VOC accepts all ANSI-74 COBOL constructs as defined in the B 4000/B 3000/B 2000 Systems COBOL ANSI-74 Reference Manual, with the exception of the continuation indicator. However, some COBOL constructs are ignored by RP3VOC. Below is a synopsis of COBOL language structure, with comments indicating whether the constructs are:

1. Recognized by RP3VOC.
2. Not recognized by RP3VOC.
3. Not applicable to data definition.
4. Accepted when defining an <input procedure>.

COBOL LANGUAGE CONSTRUCTS

RP3VOC recognizes only one data definition per line of source code. The following are the COBOL language constructs recognized by RP3VOC:

1. PRE-IDENTIFICATION DIVISION

   This division is not applicable to data definition.

2. IDENTIFICATION DIVISION

   This division is not applicable to data definition.

3. ENVIRONMENT DIVISION

   a. CONFIGURATION SECTION

      This section is not applicable to data definition.

   b. INPUT-OUTPUT SECTION

      1) FILE-CONTROL

         The entire SELECT statement is accepted in defining an <input procedure>. RP3VOC recognizes the following SELECT clauses:

         a) ASSIGN

            The hardware device name in the ASSIGN clause may not exceed 12 characters, and may not contain optional sub-clauses or more than one word.

         b) COPY

            The COPY statement is recognized by RP3VOC, including FROM...TO; however, RP3VOC ignores the REPLACING clause. The COPY statement must begin on a new line.

The following are examples of COPY statements accepted by RP3VOC:

```
COPY "PROG1".

COPY "PROG1" ON "TROPER".

COPY "PROG1" ON "TROPER" FROM 100 TO 300.
```

In this example of a COPY statement, RP3VOC ignores the REPLACING clause.

```
COPY "PROG2" REPLACING NAME BY WS-NAME.
```

The following example results in an error because the COPY statement does not begin on a new line.

```
FD INV-FILE COPY "PROG1".
```

The blocking factor should be given when you use CANDE EDITOR to create library files. For example, to copy an entire EDITOR file named LB01 to pack TROPER with the name LIB001, use an EDITOR command as shown:

```
COPY ALL OF LB01 TO PACK TROPER AS LIB001 9
```

The 9 in the previous example is the blocking factor. For more information on library files, refer to the COPY statement in the B 2000/B 3000/B 4000 Series COBOL ANSI-74 Reference Manual.

c) ORGANIZATION

d) RECORD KEY

Any qualifiers given for the RECORD KEY are ignored. The data item referenced as the key must be unique within the record description entries for the file.

RP3VOC recognizes, but does not process the following clauses:

a) ACCESS MODE
b) FILE STATUS

Sort file and all other clauses are either not recognized by RP3VOC or not applicable to data definition.

2) I-O-CONTROL

I-O-CONTROL is not recognized by RP3VOC.

4. DATA DIVISION

a. FILE SECTION

1) FILE DESCRIPTION

The entire FD is recognized by RP3VOC, with the exception of LINAGE and CODE-SET. SDs are not applicable to data definition. All options under FILE DESCRIPTION are accepted in defining an <input procedure>.

The "VALUE OF FILENAME IS <data name>" clause is accepted in FDs supplied to RP3VOC. You must supply the FILENAME value in the specifications using either: 1) the WITH ID clause of the RP3VOC FILE statement, or 2) an <input procedure>.

2) RECORD DESCRIPTION

Except for the COPY...REPLACING clause of the COPY statement, all COPY clauses are recognized by RP3VOC. The COPY statement must begin on a new line. Refer to FILE-CONTROL for more information on the COPY statement.

The 66 RENAMES is not recognized by RP3VOC. The entire 88 condition name is recognized by RP3VOC.

b. DATA-BASE SECTION

The DATA-BASE SECTION is not applicable to data definition, but it is accepted by RP3VOC for defining the <input procedure>.

c. WORKING-STORAGE SECTION

1) 77-LEVEL ITEMS

The 77-LEVEL ITEMS are accepted by RP3VOC within the <input procedure>, but are not reportable in report specifications.

2) 01-LEVEL ITEMS

The 01-LEVEL ITEMS are recognized by RP3VOC for <input procedure>s only and have the same limitations described under the heading RECORD DESCRIPTION. Refer to the COPY statement under FILE-CONTROL for additional restrictions.

If present, DEBUG lines are not included and a warning is given.

5. PROCEDURE DIVISION

This division is not applicable to data definition, but it is accepted by RP3VOC when defining an <input procedure>. Refer to the COPY statement under FILE-CONTROL for additional restrictions.

If present, DEBUG lines are not included and a warning is given.

APPENDIX D

UNRECOGNIZED COBOL CONSTRUCTS B 1000 SERIES OF SYSTEMS

The RP3VOC portion of the REPORTER III System analyzes user-supplied ANSI-74 COBOL source code. This code provides data definitions for elements being included in a vocabulary.

RP3VOC accepts all ANSI-74 COBOL constructs as defined in the B 1000 Systems COBOL74 Reference Manual, with the exception of the continuation indicator. However, some COBOL constructs are ignored by RP3VOC. Below is a synopsis of COBOL language structure, with comments indicating whether the constructs are:

1. Recognized by RP3VOC.
2. Not recognized by RP3VOC.
3. Not applicable to data definition.
4. Accepted when defining an <input procedure>.

COBOL LANGUAGE CONSTRUCTS

RP3VOC recognizes only one data definition per line of COBOL source code. The following are the COBOL language constructs recognized by RP3VOC:

1. PRE-IDENTIFICATION DIVISION

   This division is not applicable to data definition.

2. IDENTIFICATION DIVISION

   This division is not applicable to data definition.

3. ENVIRONMENT DIVISION

   a. CONFIGURATION SECTION

      This section is not applicable to data definition.

   b. INPUT-OUPUT SECTION

      1) FILE-CONTROL

The entire SELECT statement is accepted in defining an <input procedure>. RP3VOC recognizes the following SELECT clauses:

a) ASSIGN

The hardware device name in the ASSIGN clause may not exceed 12 characters, and may not contain optional sub-clauses or more than one word.

b) COPY

The COPY statement is recognized by RP3VOC, including FROM...TO; however, RP3VOC ignores the REPLACING clause. The COPY statement must begin on a new line.

Examples of COPY statements accepted by RP3VOC:

```
COPY "INFILE".

COPY "RECDES/PROG1 ON MYPACK".

COPY "INFILE ON MYPACK"
FROM 1200 TO 1400.
```

In the following example of a COPY statement, the REPLACING clause is ignored.

```
COPY "LIB1 ON DEVEL"
   REPLACING NAME BY REC-NAME.
```

The following example results in an error because the COPY statement does not begin on a separate line.

```
FD INV-FILE COPY
"MASTER/INVENTORY ON MYPACK".
```

c) ORGANIZATION

d) RECORD KEY

Any qualifiers given for the RECORD KEY are ignored. The data item referenced as the key must be unique within the record description entries for the files.

RP3VOC recognizes but does not process the following clauses:

a) ACCESS MODE
b) FILE STATUS

All other clauses are either not recognized by RP3VOC or not applicable to data definition.

2) I-O-CONTROL

I-O-CONTROL is not recognized by RP3VOC.

4. DATA DIVISION

a. FILE SECTION

1) FILE DESCRIPTION

RP3VOC recognizes FILE DESCRIPTIONS (FD) except for the following clauses:

a) LINAGE
b) CODE-SET

See COPY statement restrictions under FILE-CONTROL.

SORT-MERGE FILE DESCRIPTIONS (SD) are not applicable to data definition.

All options under FILE DESCRIPTION (FD) are accepted in defining an <input procedure>.

The "VALUE OF TITLE IS <data name>" clause is accepted in file descriptions supplied to RP3VOC. You must supply the TITLE value in the specificatons using either: 1) the WITH ID clause of the FILE statement, or 2) an <input procedure>.

2) RECORD DESCRIPTION

Except for the COPY...REPLACING clause of the COPY statement, all COPY clauses are recognized by RP3VOC. The COPY statement must begin on a new line. See the COPY statement under FILE-CONTROL for additional restrictions.

The 66 RENAMES is not recognized by RP3VOC. The entire 88 condition name is recognized by RP3VOC.

b. DATA-BASE SECTION

The DATA-BASE SECTION is not applicable to data definition, but it is accepted by RP3VOC for defining the <input procedure>.

c. WORKING-STORAGE SECTION

1) 77-LEVEL ITEMS

The 77-LEVEL ITEMS are accepted by RP3VOC within the <input procedure>, but are not reportable in report specifications.

2) 01-LEVEL ITEMS

The 01-LEVEL ITEMS are recognized by RP3VOC for <input procedure>s only and have the same limitations described under the heading RECORD DESCRIPTION. For additional limitations, refer to the COPY statement under FILE-CONTROL.

If DEBUG lines are present, they are not included and a warning is given.

d. COMMUNICATION SECTION

This section is not applicable to data definition.

5. PROCEDURE DIVISION

The PROCEDURE DIVISION is not applicable to data definition, but it is accepted by RP3VOC in defining the <input procedure>. See the COPY statement under FILE-CONTROL for limitations.

RP3VOC ignores the USING clause of the PROCEDURE DIVISION.

If DEBUG lines are present, they are not included and a warning is given.

APPENDIX E

DEFAULTS AND LIMITS

This appendix lists default values, limits, and general comments for reference when using RP3VOC.

QUALIFICATION

You can use a maximum of three name qualifiers on all systems.

SUBSCRIPTED DATA NAMES

You can use a maximum of three subscripts. The maximum values for a subscript are as follows:

System	Cobol Maximum Value	DMS II Maximum Value
A Series of Systems	65535	1023
B 2000/B 3000/B 4000 Series of Systems	999999	1023
B 1000 Series of Systems	999999	1023

EXTERNAL FILE NAMES

The maximum length of an identifier in an <external file name> is:

Characters	System
17	A Series of Systems
6	B 2000/B 3000/B 4000 Series of Systems
10	B 1000 Series of Systems

In addition, on the A Series of Systems, you are allowed a maximum of 14 identifiers in an <external file name> as long as the total length does not exceed 30 characters (including "/").

On the B 1000 Series of Systems, you are allowed a maximum of two identifiers in the <external file name>. (This does not apply to the ON clause.)

EDITING PICTURES

The maximum length of an editing picture is 30 characters. The maximum number of decimal digits represented is 18 for the B 1000 systems, and the maximum number is 23 for the A Series and B 2000/B 3000/B 4000 systems. (For default editing pictures, refer to the discussion on editing pictures in Section 2.)

STATEMENTS

The following are defaults and limits for the RP3VOC language statements:

Statement	Default/Limit
PASSWORD Statement	The maximum length of a password is 10 characters.
USER-NAME Statement	The maximum length of the <user name> is 30 characters, but only the first 20 characters are used.
SET Statement	The default blocking factor is 10 for SORT-FILE, 1 for LOGICAL-PAGE-FILE.
REPLACE Statement	The maximum number of nested macros is 10. The maximum length of each formal parameter is 30 characters with a maximum of 10 formal parameters.
DMS II Statements	The maximum length of a data base <item name> is 17 characters. The maximum length of an <alias name> is 17 characters. The population default is 9999. The default editing picture is obtained from the data-item description stored in the A Series and B 2000-B 4000 Series data-base INVOKE

	ALL listing or the B 1000 Series of Systems disjoint data set COBOL library files.
System File Statements	The total population default is 9999. The maximum length of each condition-addition literal value is 30 characters.
Input Procedures	You can use an unlimited number of <input procedure>s. However, only 20 structures may be specified in the USES clause.

APPENDIX F

RP3VOC RESERVED WORDS

Below is a list of all RP3VOC reserved words. Be careful when using these words as names. In certain contexts, the names may be confused with reserved words and cause syntax errors.

In the following list, some reserved words have the first few letters underlined. Any abbreviated form of the word that includes the underlined letters is interpreted as a reserved word. For example, "COM" and "COMMEN" are both valid versions of "COMMENTS".

ALL
AND
ANSI74
ARE
AS
ASSIGN

BASE
BY

CARDS
COBOL74
COBOL85
COMMENTS
CONTROL

DATA
DATASET
DB
DB INVOKE
DISK
DISKPACK
DYNAMICALLY

EDITING
END
END-COMMENTS
EXCLUDE

FILE
FILES
FOLLOWS
FROM

GO
GROUP

ID
IDENTIFICATION
INPUT
INTERCHANGE
IS
ITEM

LANGUAGE
LIBRARY
LOGICAL-PAGE-FILE

NOTES

OCCURRENCES
OF
ON

PACK
PACK-ID
PACKNAME
PASSWORD
PICTURE
PROCEDURE
PURGE

RANDOMLY
RECORDS
REDEFINE
REFERS
RELATES
REPLACE
ROUTINE

SEQUENTIALLY
SET
SORT-FILE

SOURCE
SPECIFICATIONS
SUBSET
SYNTAX

TAPE
THROUGH
THRU
TO
TOTAL-POPULATION

USAGE
USER-NAME
USES

VALUES
VOCABULARY
RP3VOC

WITH

APPENDIX G

REPORTER LANGUAGE RESERVED WORDS

Be careful when using REPORTER reserved words as names. In certain contexts these names may be confused with reserved words, and cause syntax errors.

If you use a reserved word as a <macro name>, the word is no longer recognized as a reserved word. If you use a key word as a <macro name>, the key word function becomes inoperative. If a vocabulary name is identical to a REPORTER reserved word, the name may be renamed via an extension (ACCT-AGE IS REDEFINITION OF AGE).

In the following list, some reserved words have the first few letters of the word underlined. Any abbreviated form of the word that includes the underlined letters is interpreted as a reserved word. For example, "ASC" and "ASCEND" are both valid versions of "ASCENDING".

ABBREVIATED
ABSTRACT
ACCEPT
ACCEPTED-DATA
AGE
ALL
AND
APPROXIMATELY
AREAS
AS
ASCENDING
ASSIGN
AT
AVERAGE
AVG

BACKUP
BASE-DATE
BL
BLANK
BLOCK-CONTAINS
BUILD
BY

CALENDAR-DATE
CENTURY-DATE
CHANGES
CHANNEL
COBOL
COLUMN-SPACING
COMBINE
COMPILE
CONTIGUOUS
COUNT

DATE
DAYS
DDDDD-DATE
DDMMMYY-DATE
DDMMYY-DATE
DECREASES
DELIMITED
DESCENDING
DEVICE
DISK
DISKPACK
DISPLAYING
DUPLICATES
DUPS

EACH
ELSE
ENTRY
EQL
EQUALS
EVERY

EXECUTION
EXTRACT
EXTRACT-VOCABULARY
EXTRACT-RP3VOC

FALSE
FILE
FINAL
FOLLOWS
FOOTING
FOOTNOTE
FOR
FORMS
FORM-LENGTH
FORM-NUMBER
FORM-PATTERNS
FORM-OVERFLOW
FORM-TYPE
FORM-WIDTH
FRACTION-SIZE
FROM
FULL-LENGTH

GENERATION
GEQ
GLOBAL-DATA
GREATER
GREGORIAN-DATE
GROUP
GROUPING
GTR

HARDWARE
HEADER

IDENTIFICATION
IF
IN
INCLUDING
INCREASES
INFORMATION
INPUT
INTEGER-SIZE
INTERCHANGE
INTO
IS

JULIAN-DATE

KEY
KEY-DATA

LEQ
LESS
LINES
LINE-NUMBER
LINE-OVERFLOW
LIST
LISTING
LOGICAL-PAGE-FILE
LOWER-BOUNDS
LSS
LT

MASTER
MATCHING
MAXIMUM
MEAN-SQUARES
MILITARY-DATE
MINIMUM
MMDDYY-DATE
MOD
MODE
MONTH
MONTHS

NAME
NEQ
NEXT
NEW
NF
NL
NOT
NOTING
NULL
NULL-BOOLEAN-ITEMS
NULL-NUMERIC-ITEMS
NULL-STRING-ITEMS
NUMERIC

OBJECT
OCCURRENCES
OF
OFF
ONLY
OR
ORDER
OTHERS
OUT
OVERFLOW
OVERFLOW-NUMBER

PACK-ID
PACKNAME

PAGES
PAGE-LENGTH
PAGE-NUMBERS
PAGE-OVERFLOW
PAGE-WIDTH
PARAMETERS
PASSWORD
PATTERN
PERCENT
PICTURE
POSITION
POSITIONS
PREFIXED
PRESELECT
PRINT
PRINTER
PROCESSING
PUNCH

QUARTERS
QUOTES

RANDOMLY
RANGE
READER
RECORDS
REDEFINITION
RELATED
REPLACE
REPORTS
ROUNDED
RUNNING-COUNT
RUNNING-TOTAL

SAMPLED
SAVE
SEED
SELECT
SET
SIZE
SORT-FILE
SOURCE
SPACES
SPECIFICATIONS
SPO
STANDARD
START-POINT
STARTING
STD-DEVIATION
STRATA
STRING
STRING-SIZE
SUFFIXED
SUM-SQUARES
SUMMARIZE
SUMMARY
SUPPRESS
SYSTEMATICALLY

TABLE
TAPE
TERMINAL
TEXT
THAN
THEN
TIME
TITLE
TO
TOP-OF-FORM-CHANNEL
TOTAL
TOTAL-POPULATION
TRUE

UNMATCHED
UNSORTED
UNTIL
UPPER-BOUNDS

VALUE
VARIANCE
VERTICAL-SPACING
VIA
VOCABULARY

WEEKS
WHERE
WHICH
WITH
WITHOUT

YEARS
YYDDD-DATE
YYMMDD-DATE

ZEROS

APPENDIX H

ERROR MESSAGES

This appendix explains error or warning messages which are issued by the RP3VOC program and appear on the output listing of RP3VOC specifications.

Error (or warning) messages immediately follow the line in which the error occurs. The numbers to the left of the error message in the output listing correspond to the number in the left column of this documentation.

If an asterisk appears in an error message, it appears directly beneath the expression causing the error, or beneath the adjacent expression.

000 (not in use)

001 MISSING 'IS' OR '='

The word "IS" or "=" was expected at this point in the statement but was not given.

002 COBOL WORD EXPECTED

A COBOL word from the RP3VOC specifications was expected, but none was given.

003 ILLEGAL PASSWORD

The password must be a <string> or <word> of 10 characters or less.

004 PICTURE TOO LONG

The maximum length of the picture is 30 characters.

005 COBOL NAME NOT IN THIS FD

The name given in the RP3VOC specifications was not found in the particular file description of the COBOL source file. Names within a file description or SD in an <input procedure> cannot be referenced or changed.

006 MISSING KEYWORD: 'VALUES'

"VA", "VALUE", or "VALUES" is needed following the word "WITH" in the condition-addition statement of the RP3VOC specifications.

007 ILLEGAL CONDITIONAL VALUE

A <number> or <string> must follow "VA", "VALUE", "VALUES", "THRU", or "THROUGH" in the condition-addition statement.

008 FD NOT IN THIS FILE

The file named in the FILE statement of the RP3VOC specifications was not found in the COBOL source file. Since RP3VOC accesses items in the COBOL source file sequentially, the FDs must be referenced in the sequence presented in the source file.

009 MISSING PERIOD: '.'

A period was expected at this point, but none was found.

010 INCORRECT HARDWARE DEVICE SPECIFICATION

The hardware medium upon which the COBOL source file resides has been incorrectly specified. The valid media are disk, tape, card, disk pack, or library. COBOL source may also be listed as immediately following the SOURCE statement in the RP3VOC specifications.

011 MISSING QUALIFIER

A <name> is expected following the word "OF" or "IN" in the statement.

012 COMMA "," OR ACCESS METHOD EXPECTED

The word SEQUENTIALLY (SEQ), RANDOMLY (RANDOM), DYNAMICALLY (DYN), or a comma (,) was expected at this point, but none was found. This might happen if SEQUENTIALLY, RANDOMLY, or DYNAMICALLY was misspelled.

For A Series Systems only:
013 THE DEFAULT STORAGE PICTURE OF <storage picture> WILL BE USED FOR <data name>. REPORTING ON THIS ITEM MAY RESULT IN ERRORS.

RP3VOC found a data name with an unrecognizable storage picture so one will be assigned. An example of this would be an item in a data base which was declared as a REAL but the size was not given. Reporting on this item may result in syntax errors in the generated source program depending on the level of the COBOL compiler, or if the program compiles possible truncation of data may result.

014 FORTE & FORTE/2 ARE NOT SUPPORTED.

A FORTE or FORTE/2 structure was found in the vocabulary specification. These structures are not supported in ANSI-74 COBOL.

015 'TO' EXPECTED

The word "TO" was expected at this point in the statement but was not given.

016 BOTH VOCABULARY FILES HAVE THE SAME NAME

The two <external file name>s in the VOCABULARY statement are used to identify two files that contain the vocabulary. The two <external file name>s should be different.

017 'USAGE ANSI-74' IS NOT REQUIRED.

The clause "USAGE ANSI-74" was found in the vocabulary specification. This is a warning that the clause has no effect since only ANSI-74 COBOL is used.

018 ILLEGAL RP3VOC STATEMENT

This statement is not a legal statement in the vocabulary language (RP3VOC).

For A Series Systems only:
019 INVALID LANGUAGE SPECIFIED. EXPECTING EITHER 'COBOL74' OR 'COBOL85'

An invalid language name was specified in the SET OPTION statement. The valid COBOL languages are COBOL74 and COBOL85.

020 (not in use)

021 ILLEGAL OPTION SPECIFIED

This can be either a warning message or an error message. It is a warning if the option statement in the RP3VOC specification contains an illegal option or an unrecognized construct. This message is an error if an illegal option is specified in the SET statement.

022 PASSWORD IS TOO LONG

A password can contain no more than 10 characters.

023 MISSING KEYWORD: 'PROCEDURE'

"PROC", "PROCEDURE", or "ROUTINE" was expected after the word "INPUT" in the RP3VOC INPUT PROCEDURE statement.

024 (not in use)

025 NAME WITH QUALIFICATION EXCEEDS 60 CHARACTERS

A data name with its associated qualifiers can total no more than 60 characters.

026 KEYWORD "WITH" EXPECTED

If no period follows the new <data name> in a data-name-change statement, RP3VOC expects the word "WITH" followed by a picture clause.

027 MISSING FILE-CONTROL OR DATA DIVISION

A FILE-CONTROL or DATA-DIVISION was not found in the COBOL source file.

028 ':' EXPECTED

A colon (:) was expected at this point in the statement, but none was given.

029 (not in use)

030 KEYWORD 'AND' EXPECTED

The VOCABULARY statement requires two file names separated by the word "AND".

031 NOT VALID COBOL SYNTAX

Invalid syntax in the COBOL source file. Recompile the COBOL source file to ensure an error-free source file.

032 MISSING END FILE STATEMENT OR FILE STATEMENT SPECIFIED IS OUT OF CONTEXT

Every FILE statement must have an accompanying END FILE statement. This error occurs in an <input procedure> if the FILE statement or END FILE statement should not have been included.

033 A VALID FILE STATEMENT MUST PRECEDE THIS

The current RP3VOC statement must occur within a FILE/END FILE block.

For B 2000/B 3000/B 4000 Systems:
034 THE WORD 'PACK' WAS EXPECTED

The word 'PACK' was expected in the statement DDF ON PACK "<pack ID>", but it was not found.

035 INVALID EXTERNAL-FILE-NAME

An invalid format for an <external file name> for this hardware device has been specified.

036 INTEGER EXPECTED

An integer was expected at this point in the statement, but none was given.

037 INVALID STATEMENT -- MAY CAUSE FURTHER ERROR

The key word does not introduce a valid RP3VOC statement. The statement will be ignored.

038 STRING OR WORD EXPECTED

A <string> or <word> was expected at this point in the statement, but none was given.

039 INPUT PROCEDURE MUST HAVE AN INPUT-ROUTINE

The user-written <input procedure> must have a paragraph labelled "INPUT-ROUTINE".

040 INPUT PROCEDURE STATEMENT MUST PRECEDE THIS

An END PROCEDURE statement has been encountered without the corresponding INPUT PROCEDURE or INPUT ROUTINE statement.

041 MISSING PROCEDURE DIVISION

"PROCEDURE DIVISION" is missing from the COBOL source file.

042 'END PROCEDURE' EXPECTED

"END PROCEDURE" is required to match the corresponding "INPUT PROCEDURE" statement. None was encountered.

043 STRING EXPECTED

A <string> was expected at this point in the statement, but none was given.

044 INVALID COBOL SYNTAX -- RESULTS UNPREDICTABLE

Refer to the appropriate Unisys ANSI-standard COBOL reference manual.

045 STRING IS TOO LONG

The <string> is too long.

046 EXTERNAL-FILE-NAME IS TOO LONG

On A Series Systems, the maximum length of a particular file identifier is 17 characters. The maximum length for an <external file name> is 30 characters, including quotes and slashes (not applicable to the ON clause). On B 2000/B 3000/B 4000 Systems, the maximum length of a file identifier is six characters. On B 1000 Systems, the maximum identifier is 10 characters.

047 COBOL WORDS CANNOT EXCEED 30 CHARACTERS

A COBOL word was found that exceeded 30 characters in length.

048 (not in use)

049 NO SELECT GIVEN FOR THIS FILE OR SELECT GIVEN WAS NOT VALID COBOL SYNTAX

No SELECT statement was found for this file in the COBOL source file, or the SELECT statement contained unrecognized COBOL74 or COBOL85 syntax.

050 USING NUMERIC EDITED ITEMS IN REPORTS WILL CAUSE COBOL SYNTAX ERRORS

Data names processed by RP3VOC that have numeric edited pictures (characters "Z", "*", ".", ",", "S", "$") must not be used in the report syntax. Doing so results in syntax errors during the compilation of the generated COBOL74 or COBOL85 program.

For B 2000/B 3000/B 4000 Systems:
051 INVALID PACK-ID

The PACK-ID name exceeded six characters.

052 BAD DB INVOKE RECORD

A bad record was detected in the fixed-format DB INVOKE/<data-base name> (interface) file.

053 SET-NAME EXPECTED

A valid <set name> was expected in the DMS II SET statement.

054 ITEM NOT FOUND

The referenced item does not exist (currently applies to DMS II items).

055 INVALID ASSIGN CONSTRUCT

RP3VOC does not recognize the word following "ASSIGN" in the DMS II ASSIGN statement.

056 ITEM NAME EXPECTED

A name is expected in the DMS II statement in the RP3VOC specifications.

057 MULTIPLE DATASET STATEMENTS FOR SAME DISJOINT DATA SET

A disjoint data set may be referenced by only one DATASET statement. Additional references must be made by REDEFINE statements.

058 WORD 'AS' EXPECTED

The word "AS" was expected in the DMS II statement.

For B 1000 Systems:
059 UNRECOGNIZED DMS II CONSTRUCT IN COBOL LIBRARY FILE <NAME>

RP3VOC was unable to identify a DMS II construct in the COBOL library file with the name given.

For A Series and B 2000/B 3000/B 4000 Systems:
059 UNRECOGNIZED DMS II CONSTRUCT IN INVOKE SOURCE FILE

RP3VOC was unable to identify a DMS II construct in the DB INVOKE source file.

For B 1000 Systems:
060 UNEXPECTED END OF COBOL LIBRARY FILE

The end of the COBOL library file was encountered before all required data-base information was found.

For A Series and B 2000/B 3000/B 4000 Systems:
060 UNEXPECTED END OF INVOKE SOURCE FILE

The end of the DB INVOKE file was encountered before all required data-base information was found.

For A Series and B 2000/B 3000/B 4000 Systems:
061 ERRORS IN INVOKE SOURCE FILE. SKIPPING TO END DB CLAUSE

Errors detected in the DB INVOKE file. All RP3VOC data-base information specifications present were ignored.

For A Series and B 2000/B 3000/B 4000 Systems:
062 DATA BASE SECTION NOT PRESENT IN INVOKE SOURCE

"DATA-BASE SECTION" was not found in the DB INVOKE file.

063 HIGHER DATASET NOT SEEN IN 'USES' CLAUSE

An embedded data set has been specified in the USES clause in a user-written <input procedure>; however, the owner data set has not been previously specified in the USES clause.

064 PROGRAM INTERNAL ERROR. ENTERED DM-VOCAL-INTERFACE WITH INTERFACE OPEN.

This error applies to the A Series systems only. It indicates the vocabulary language program had an internal error and the DM-VOCAL-INTERFACE paragraph is/was being performed with the INTERFACE file open. If this error occurs, all the specification files and all additional files needed to reproduce the problem should be submitted to Burroughs for analysis.

065 LOGICAL DATABASE INCORRECTLY DECLARED AS PHYSICAL IN RP3VOC SPECS.

The DMS II data base declared in the <data-base name> of the DB statement in the RP3VOC specs is actually a logical data base.

066 UNRECOGNIZED ITEM TYPE

Unrecognized item type.

067 NAME EXPECTED

A <name> was expected at this point in the statement, but none was given.

068 WORD 'DB' EXPECTED

The word "DB" was expected in the RP3VOC specifications, but none was found.

069 RP3VDM ABORTED

No DB INVOKE file was present. Therefore, RP3VOC unsuccessfully attempted to run RP3VDM (A Series of Systems).

For A Series and B 2000/B 3000/B 4000 Systems:
070 DASDL DESCRIPTION FILE IS NOT ON DISK

No DB INVOKE file was present and one could not be generated because the DASDL description file is missing.

For A Series and B 2000/B 3000/B 4000 Systems:
071 RP3VDM NOT ON DISK

No DB INVOKE file was present and one could not be generated because RP3VDM was missing.

For A Series and B 2000/B 3000/B 4000 Systems:
072 I/O ERROR IN DB INVOKE FILE

A read of the DB INVOKE file terminated with a condition other than End-of-File.

For A Series and B 2000/B 3000/B 4000 Systems:
073 DB INVOKE FILE SOURCE FILE IS EMPTY

The DB INVOKE source file accessed by RP3VOC did not contain any information.

074 DATA SET NAME EXPECTED

A <name> given in a DATASET statement or a REDEFINE statement does not reference a data set.

075 REDEFINE STATEMENT DOES NOT REFERENCE LAST DISJOINT DATA SET

The REDEFINE statement must come after the specified data set has been processed.

076 EMBEDDED SETS CANNOT BE GIVEN AN ALIAS

A SET statement references an embedded set, which makes an AS clause illegal.

077 ONLY ONE DATABASE ALLOWED IN INPUT

Information from one, and only one, data base may be reported.

078 DATA-NAME WILL BE UNDEFINED IN SYNTAX CHECK; USER INPUT
PROCEDURE MUST DEFINE AND SET DATA-NAME

This warning occurs only if the WITH ID clause of the FILE statement is not in the RP3VOC specifications, and a file description contains either a:

a) "VALUE OF TITLE IS <data name>" clause (B 1000 and A Series of Systems), or

b) "VALUE OF FILENAME IS <data name>" or "VALUE OF FAMILYNAME IS <data name>" clause (A Series and B 2000/B 3000/B 4000).

Under these conditions the RP3VOC syntax check produces an undefined symbol error for <data name>. This should be ignored if the data name is defined in the WORKING-STORAGE SECTION of an <input procedure>. The input procedure should move a data value into <data name> before the file is opened. For files not assigned to an input procedure, you can use the WITH ID clause of the FILE statement to assign a name.

079 DATA STRUCTURE NOT PREVIOUSLY SPECIFIED TO RP3VOC

In a user-written <input procedure>, a system file or DMS II data set declared in the USES clause has not been previously defined to RP3VOC.

080 (not in use)

081 LIMIT ON DATA STRUCTURES EXCEEDED

The user-written <input procedure> may have a maximum of five data structures specified in the USES clause of the <input procedure> statement.

082 MISSING END-OF-JOB. (MUST START IN COL. 8). JOB
ABORTED.

The input routine source was missing an END-OF-JOB paragraph, or it did not begin in column 8.

083 '#' NOT FOUND FOR MACRO

The number sign (#) marks the end of each macro definition in the REPLACE statement. Since no number sign was found, all the subsequent RP3VOC specifications were assumed to be part of the macro text.

084 (not in use)

085 MACRO ERROR

This is a general error in the REPLACE statement. The following are the specific error messages which can arise:

DUPLICATE FORMAL PARAMETER NAME

The parameters named in the parameter list must be unique among themselves.

IDENTIFIER PREVIOUSLY DECLARED

The <macro name> must be unique.

ILLEGAL IDENTIFIER

Integers, strings, and special characters may not be used as formal parameters or <macro name>s; only words may be so used.

ILLEGAL MACRO INVOKE

A macro cannot be used to define itself, for example:

```
REPLACE A BY A + B#.
```

MACRO DEFINE LIMIT EXCEEDED

Macros may only be defined to a level 10-deep.

MISSING KEYWORD - 'BY'

"BY" must separate the <macro name> from the macro text.

MISSING RIGHT PARENTHESIS - ')'

Left and right parenthesis must correspond. In particular, a right parenthesis must mark the end of the list of formal parameters.

TOO MANY FORMAL PARAMETERS

A maximum of 10 formal parameters is permitted.

086-88 (not in use)

089 THE WORD 'IS' WAS EXPECTED

The word 'IS' was expected in the statement SET DECIMAL-POINT IS COMMA, but it was not found.

090 THE WORD 'COMMA' WAS EXPECTED

The word "COMMA" was expected in the statement SET DECIMAL-POINT IS COMMA, but it was not found.

091 A PERIOD WAS EXPECTED

A period terminating the statement SET DECIMAL-POINT IS COMMA was expected but not found.

092-104 (not in use)

105 VA ILLEGAL IN ANSI-74 AND ANSI-85 COBOL. CHANGED TO VALUE. |

Since the word VALUE rather than VA is preferred and in some cases required, RP3VOC changed VA to VALUE.

106 PC ILLEGAL IN ANSI-74 AND ANSI-85 COBOL. CHANGED TO PIC. |

Since the word PIC rather than PC is preferred and in some cases required, RP3VOC changed PC to PIC.

107 OC ILLEGAL IN ANSI-74 AND ANSI-85 COBOL. CHANGED TO OCCURS.

Since the word OCCURS rather than OC is preferred and in some cases required, RP3VOC changed OC to OCCURS.

108 USAGE TYPE ILLEGAL IN ANSI-74 AND ANSI-85 COBOL.

One of the following picture usage types was seen in a COBOL74 source file or in a COBOL85 source file:

ASC11
CMP
CMP-1
CMP-3
COMP-2
COMP-3
COMP-4
COMP-5
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTATIONAL-3
COMPUTATIONAL-4
COMPUTATIONAL-5
DISPLAY-1

The above usage types are COBOL68. The user must make appropriate changes to the COBOL74 source file or COBOL85 source file to ensure that the picture and usage type given match the data field in the file.

For B 1000 Systems:
109 IDENTIFICATION ILLEGAL IN ANSI-74 COBOL. CHANGED TO TITLE.

While examining the file, the clause VALUE OF IDENTIFICATION was found. This is COBOL68 syntax. The word IDENTIFICATION was changed to TITLE to conform with COBOL74 syntax.

For A Series and B 2000/B 3000/B 4000 Systems:
109 IDENTIFICATION ILLEGAL IN ANSI-74 AND ANSI-85 COBOL. CHANGED TO FILENAME.

While examining the file, the clause VALUE OF IDENTIFICATION was found. This is COBOL68 syntax. The word IDENTIFICATION was changed to FILENAME to conform with COBOL74 or COBOL85 syntax as appropriate. Note that COBOL85 is available only on A Series Systems.

110 (not in use)

111 PICTURE FOR THIS ITEM EXCEEDS MAXIMUM NUMBER OF DIGITS

A maximum number of 18 digits is allowed in numeric picture clauses for B 1000 Systems by the ANSI-74 COBOL compiler. The printed item's numeric picture clause has exceeded this limit.

A maximum number of 23 digits is allowed in numeric picture clauses for B 2000/B 3000/B 4000 Systems by the ANSI-74 COBOL compiler. The printed item's numeric picture clause has exceeded this limit.

A maximum number of 23 digits is allowed in numeric picture clauses for A Series Systems by the ANSI-74 and ANSI-85 COBOL compilers. The printed item's numeric picture clause has exceeded this limit.

For A Series and B 2000/B 3000/B 4000 Systems:
112 INVOKE SOURCE ERROR. PERIOD NOT FOUND.

An error was detected in the DB INVOKE file. A period was expected but was not found.

For A Series and B 2000/B 3000/B 4000 Systems:
113 "DB <DB-NAME>" FOR THIS DATABASE NOT FOUND

A DMS II data-base invocation statement was not found in the DB INVOKE file for this data base.

114 DATABASE NAME MUST BE 3 OR MORE CHARS.

A B 2000/B 3000/B 4000 series DMS II data-base name must be at least three characters long.

115 UNEXPECTED DATABASE. IGNORED.

The data-base name found in the DB INVOKE file does not match the <data-base name> specified in the DB statement. The data-base name found in the DB INVOKE file is ignored.

116 NEW VERSIONS OF DB INVOKE FILES ARE BEING CREATED

No file with either the name DB/INVOKE/<data base>, or the name DB/INVOKE/<data base>/<logical data base> was found. RP3VDM will be run to create the file. A new DB-INVOKE file was created (A Series).

117 MAXIMUM NUMBER OF SUBSCRIPTS EXCEEDED - LIMIT IS 3

A maximum of three subscript levels is allowed for a subscripted <data name>.

118 INCOMPATIBLE RELEASE VERSION

The vocabulary is incompatible with the current release version of RP3VOC.

119 EMPTY LIBRARY FILE ENCOUNTERED

A library file with no COBOL statements in it was encountered during the processing of a COPY statement.

120 INVALID COBOL LEVEL NUMBER

A data item with an invalid number was encountered. The RP3VOC specification should be rerun using the LIST SYNTAX option to identify these items.

121 EDITING PIC LENGTH GREATER THAN 3 DIGITS: CHANGED TO 999.

Specified "PIC 9" editing picture length was greater than 999. Editing picture has been changed to a length of 999.

122 EDITING PIC FRACTION SIZE GREATER THAN 2 DIGITS: CHANGED TO 99.

Specified fraction size of a "PIC 9" editing picture was greater than two digits. Fraction of editing picture has been changed to a length of 99.

123 EDITING PIC GREATER THAN 132 CHARACTERS. MAY RESULT IN ERRORS IN REP IF ITEM IS REPORTED ON

A total editing picture length is greater than 132 characters is allowed in RP3VOC, and may result in errors in the REPORTER III System if the item is reported on.

124 END FILE, PROCEDURE, OR ROUTINE EXPECTED

The word FILE, PROCEDURE (PROC), or ROUTINE was expected. This error will occur if END PROCEDURE or END ROUTINE is not found after processing all input routines. This might happen if PROCEDURE or ROUTINE is misspelled.

For A Series systems:
125 DB/INVOKE/<data base> IS THE INCORRECT VERSION.
PLEASE REMOVE IT AND RUN RP3VOC AGAIN.

The DB/INVOKE file was created with VOCDMI. This DB/INVOKE file cannot be used by RP3VOC.

126 (NOT IN USE)

For B 2000/B 3000/B 4000 Systems:
125 <DB-INVOKE file name> IS THE INCORRECT VERSION.
PLEASE REMOVE IT AND RUN RP3VDM THEN RP3VOC AGAIN.

The DB/INVOKE file was created with VOCDMI. This DB/INVOKE file cannot be used by RP3VOC.

126 PICTURE FOR THIS ITEM EXCEEDS REPORTER III DESIGN LIMIT.

A data name with a numeric picture greater than 23 digits in length was found. Although REPORTER III allows this data name to be used in statements such as REPORT, PRINT, and BUILD, using this data name in an extension can truncate the final result for numeric pictures that are longer than 23 digits. If truncation occurs, an exception is issued.

The following are errors and warnings not prefixed by a number.

ERROR INVALID DUPLICATE NAME IN VOCABULARY, FILES WILL BE PURGED
---<DATA-NAME>

Names given to RP3VOC which cannot legally be qualified must be unique.

ERROR MISSING VOCABULARY STATEMENT -- JOB ABORTED.

The <VOCABULARY statement> was missing or given incorrectly. This can occur if the statement did not fall within columns 8-72 of either the card deck or a CANDE file of type COBOL or data.

*** FATAL ERROR *** RP3VOC CAN NOT HANDLE BCL FILES

Only EBCDIC files are acceptable.

WARNING ILLEGAL CHARACTER IN COL. 7. LINE IGNORED.

Acceptable characters in column 7 are /, *, and L; continuation hyphens are not recognized. All other characters are considered illegal.

WARNING WE ARE CURRENTLY UNABLE TO HANDLE CONTINUATION. PLEASE REFORMAT SOURCE.

RP3VOC currently does not recognize COBOL source statements which are continued over a line through the use of a hyphen (-) in column 7. The source should be reformatted, continuations eliminated, and RP3VOC rerun.

WARNING REFERENCE MADE TO UNDEFINED USER PROCEDURE

*** SOFTWARE INCOMPATIBILITY ERROR ***

This message occurs on A Series systems. A more recent MCP version is required to run the REPORTER III System.

The next six messages are internal errors of the RP3VOC System. They are followed by the message: SEND ALL LISTINGS TO BURROUGHS.

INVALID KEY WHILE ACCESSING FD-FILE

INVALID KEY WHILE READING VOCAF

INVALID KEY WHILE READING WORKV

INVALID KEY WHILE WRITING FD-FILE

INVALID KEY WHILE WRITING VOCAF

INVALID KEY WHILE WRITING WORKV

APPENDIX I

RP3VDM ERROR MESSAGES

This appendix explains error or warning messages issued by the RP3VDM program. These messages appear on the output listing of the DB INVOKE listing.

001 EMPTY INPUT SPECIFICATION FILE

The data-base name, run-time options, or non-default file attributed to the DB INVOKE file was expected, but none was given.

002 DB EXPECTED

The word "DB" was expected at this point in the RP3VDM input specification, but none was encountered.

003 DATABASE NAME EXPECTED

A <data-base name> was expected in the DB statement in the RP3VDM input specification.

004 INVALID DATA BASE NAME

The data-base name specified had more than six characters (B 2000/B 3000/B 4000 Systems only). The data-base name specified had more than 17 characters (A Series Systems only).

005 DB LISTING NO LONGER SUPPORTED

For B 2000/B 3000/B 4000 Systems only, the DB LISTING option is no longer available.

006 INVALID LOGICAL DATABASE NAME OR PHYSICAL DATABASE NAME SPECIFIED

An invalid format for a physical-data-base name or a logical-data-base name has been specified . The logical- or physical-data-base name cannot exceed 17 characters. Valid data-base names consist of the following characters: A-Z, 0-9.

007 PHYSICAL DATABASE NAME EXPECTED

A physical-data-base name was expected at this point in the statement, but none was given.

008 INVALID RUN-TIME OPTION - LIST, SYNTAX, OR DEBUG
EXPECTED

A run-time option is expected. RP3VDM does not recognize the word; only "LIST", "SYNTAX", "COBOL74", "COBOL85", and "DEBUG" are valid. |

009 LIST OPTION PREVIOUSLY SPECIFIED

The word "LIST" was previously specified as a run-time option.

010 SYNTAX OPTION PREVIOUSLY SPECIFIED

The word "SYNTAX" was previously specified as a run-time option.

011 DEBUG OPTION PREVIOUSLY SPECIFIED

The word "DEBUG" was previously specified as a run-time option.

012 LIST, SYNTAX, OR DEBUG EXPECTED

A run-time option was expected at this point in the statement, but none was found.

For B 2000/B 3000/B 4000 Systems:
013 INVALID DATA DEFINITION FILE OR DB INVOKE FILE ATTRIBUTE
SPECIFIED

RP3VDM expects to find the option words "DDF-PACK" or DB-INVOKE."

For B 2000/B 3000/B 4000 Systems:
014 DDF-PACK PREVIOUSLY SPECIFIED

The data-definition file attribute was previously specified.

015 DB INVOKE PREVIOUSLY SPECIFIED

The DB INVOKE file attribute was previously specified.

For B 2000/B 3000/B 4000 Systems:
016 PACK-ID EXPECTED

The restricted disk pack name was expected in DDF-PACK statement.

For B 2000/B 3000/B 4000 Systems:
017 INVALID PACK-ID

The PACK-ID name exceeded six characters. B 3000/B 4000 systems only).

018 DB INVOKE FILE ATTRIBUTE SPECIFIED

The family <PACK-ID> or the <file name> of the DB INVOKE file was not specified in the DB INVOKE statement.

019 INVALID DB INVOKE FILE ATTRIBUTE SPECIFIED

An invalid format for the DB INVOKE file attribute was specified.

020 INVALID PACK-ID,"/" NOT ALLOWED IN PACK-ID

A "/" is not allowed in the character string of the PACK-ID.

021 INVALID DB INVOKE FILENAME, "/" NOT ALLOWED IN DB INVOKE FILENAME

A "/" is not allowed in the character string of the DB INVOKE file name.

For B 2000/B 3000/B 4000 Systems:
022 FILENAME TRUNCATED TO 6 CHARACTERS

The maximum length of a DB INVOKE file name is six characters.

023 INVALID RP3VDM OPTION SPECIFIED

An invalid format for RP3VDM option was specified.

024 MAXIMUM TIME ALLOWED FOR COBOL COMPILE EXCEEDED

RP3VDM will wait for a maximum of 7295 seconds (2 hours and 5 seconds) for the COBOL compile to finish.

025 DATA-BASE SECTION IN DB INVOKE FILE WAS MISSING, A PRINTOUT OF COMPILE LISTING WAS GENERATED

The data-base section in DB INVOKE file was missing. A printout of the compile listing was generated.

026 ERROR DETECTED IN COMPILER LISTING, PRINTOUT OF LISTING AS FOLLOWS:

Syntax error was found in DB INVOKE file. The printout of the compile listing was generated.

027-031 [unused]

032 COMMA EXPECTED

A comma (,) was expected at this point in the statement, but none was given.

033 TOO MANY COMMAS SPECIFIED

If a comma is seen, the following token is expected to be listed as a run-time option (LIST, SYNTAX, OR DEBUG), but a run-time option was not given.

034 PRECEDING COMMA IGNORED

If a comma follows the data-base name or file name and no run-time options are specified, a warning message is printed.

For B 2000/B 3000/B 4000 Systems:
035 DB-INVOKE <FILENAME> OR <PACK-ID>/<FILENAME> REQUIRED WHEN LOGICAL DATA BASE IS SPECIFIED

A DB-INVOKE file name is required when a logical data base is specified.

For B 2000/B 3000/B 4000 Systems:
036 TOKEN = <6-digit number> THIS IS ASSUMED TO BE A SEQUENCE NUMBER AND WILL BE IGNORED.

If the specification file is a COBOL type, the 6-digit sequence number is ignored and a warning is given.

037 COBOL74 OR COBOL85 OPTION PREVIOUSLY SPECIFIED

The word "COBOL74" or "COBOL85" was previously specified as a run-time option.

APPENDIX J

CONVERTING A SERIES COBOL74 PROGRAMS TO COBOL85 PROGRAMS

This appendix provides the information you need to upgrade your REPORTER III COBOL74 programs to COBOL85. In most cases COBOL74 report specifications can be used in a COBOL85 environment without modification. However, vocabulary specifications might require some modification. The work required depends on the location of your COBOL74 code as follows:

- If your vocabulary specifications point to existing COBOL74 applications to retrieve file descriptions, you need to alter your COBOL74 source code to make it compatible with COBOL85. Once you have converted the source code, recompile the code using the COBOL85 compiler.

  Refer to the "A Series COBOL ANSI-85 Programming Reference Manual, Volume 1: Basic Implementation" (8600 1518) for information on the differences between A Series COBOL74 and A Series COBOL85.

- If you use in-line COBOL74 code for file descriptions or input procedures, then you must inspect the code to determine the best method for creating COBOL85 code. The methods available to you are discussed in this appendix.

TRANSLATING SIMPLE IN-LINE CODE

If the in-line code is simple, or if there are only a few vocabulary specifications, refer to the "A Series COBOL ANSI-85 Reference Manual, Vol. 1" for information on translating the code.

TRANSLATING COMPLEX IN-LINE CODE

If the in-line code is extensive, or if you have used COBOL74 features extensively, you can use any of the following three methods to make your vocabulary specifications usable with COBOL85.

Method 1

Review your vocabulary specifications and as necessary manually alter them to be compatible with the COBOL85 compiler. Refer to the "A Series COBOL ANSI-85 Reference Manual, Vol. 1" for information on the differences between COBOL74 and COBOL85.

Method 2

Method 2 involves the following four steps:

1. Move the in-line code from the vocabulary specifications to a COBOL74 source file.

2. Add the other COBOL74 elements required to make a complete COBOL74 program.

3. Translate the COBOL74 program to COBOL85 using the information in the "A Series COBOL ANSI-85 Reference Manual, Vol. 1."

4. Once you have created the COBOL85 program, insert the new information into the vocabulary specification, or modify the vocabulary specification to point to the program you created.

Method 3

Method 3 uses the SYNTAX clause of the LIST statement to create the COBOL74 programs for you.

REPORTER III operates in the following way using the SYNTAX clause:

1. After RP3VOC creates the vocabulary files, it initiates RP3VGN.

2. RP3VGN creates a COBOL74 program for each file, data base, and input procedure specified in the vocabulary specification.

3. RP3VGN then calls a WFL job that initiates the COBOL74 compiler to perform a syntax compile on each of the programs.

4. As each syntax compile completes successfully, the WFL job controlling the compile automatically deletes the source file created by RP3VGN.

In order to capture the COBOL74 source files, you must prevent the WFL job from deleting them.

If you can capture the source files, then you can use the information in the "A Series COBOL ANSI-85 Reference Manual, Vol. 1" to translate the COBOL74 source files to COBOL85.

Use any of the following methods to capture the source files:

- This is the preferred method for capturing source files. However, you can only use this method if you have access to the REPORTER III source files.

  Modify the SYMBOL/RP3VGN source code file at line number 275100 to prevent RP3VGN from calling SYSTEM/WFL. Then compile SYSTEM/RP3VGN as RP3VGN using the RP3/COMPILE job file from the release tape.

  Since the WFL job is not called, the COBOL source files are not compiled and therefore they do not get deleted.

  Once you have captured the source files, replace the modified RP3VGN with the version supplied on the release tapes.

- Temporarily remove access to the COBOL74 compiler by using the FAMILY DISK statement, either in a WFL job or through CANDE. Using the statement "FAMILY DISK = <primary pack name> ONLY" limits access to only the primary pack. This pack should contain the REPORTER III object code files, the vocabulary specifications, and any external files accessed by a vocabulary specification. It should not contain the COBOL74 compiler.

  When WFL tries to initiate the compiler, the job will hang, looking for the COBOL74 compiler. At this time, you can discontinue the WFL job and save the COBOL source files.

- Stop the compiler before it finishes compiling each source file. Then copy the source files to a safe place before you restart the compiler.

  Identifying the correct time to stop the compiler might be difficult because some of the source files can be quite small.

You need to save the source files after each run of RP3VOC. For each file, use a different file name than the one created by RP3VGN. The naming convention for the source files is VCnnnn, where nnnn is a 4-digit number.

The names of the source files are not related to the name of the vocabulary specification. It is therefore important that you keep track of which source files belong with which vocabulary specification.

VERIFYING THE CONVERTED CODE

Once you have converted the COBOL74 source code to COBOL85 and modified the vocabulary specification file to point to the translated COBOL85 source code (either by including the new code in-line or by pointing to the COBOL85 source files), regenerate the vocabulary with the SET LANGUAGE TO COBOL85 and LIST SYNTAX statements. This regeneration verifies that the COBOL85 code has been correctly included in the vocabulary.

INDEX

INDEX (continued)

INDEX (continued)

<u>INDEX</u> (continued)

INDEX (continued)

INDEX (continued)

INDEX (continued)

INDEX (continued)